SESSION

WORLDCOMP/GCA KEYNOTES

Chair(s)

TBA

WORLDCOMP/GCA Keynote: Computing With Words and Perceptions—A Paradigm Shift

Lotfi A. Zadeh[*]

There are many misconceptions about what Computing with Words (CW) is and what it has to offer. A common misconception is that CW is closely related to natural language processing. In reality, this is not the case. More importantly, at this juncture what is widely unrecognized is that moving from computation with numbers to computation with words has the potential for evolving into a basic paradigm shift—a paradigm shift which would open the door to a wide-ranging enlargement of the role of natural languages in scientific theories.

In essence, CW is a system of computation which adds to traditional systems of computation two important capabilities: (a) the capability to precisiate the meaning of words and propositions drawn from natural language; and (b) the capability to reason and compute with precisiated words and propositions.

As a system of computation, a CW-based model, or simply CW-model, has three principal components. (a) A question, Q, of the form: What is the value of a variable, Y? (b) An information set, $I=(p_1, ..., p_n)$, where the p_i, $i=(1, ..., n)$, are propositions which individually or collectively are carriers of information about the value of Y, that is, are question-relevant. One or more of the p_i may be drawn from world knowledge. A proposition, p_i, plays the role of an assignment statement which assigns a value, v_i, to a variable, X_i, in p_i. Equivalently, p_i may be viewed as an answer to the question: What is the value of X_i? X_i and v_i, may be explicit or implicit. A proposition, p_i, may be unconditional or conditional, expressed as an if-then rule. Basically, an assignment statement constrains the values which X_i is allowed to take. To place this in evidence, X_i and v_i are referred to as the constrained variable and the constraining relation, respectively. More concretely, what this implies is that the meaning of a proposition, p, may be represented as a generalized constraint, X isr R, in which X is the constrained variable, R is the constraining relation and r defines the modality of the constraint, that is, the way in which R constrains X. When v_i is a word or a combination of words, X_i is referred to as a linguistic variable, with v_i being its linguistic value. When it is helpful to stress that p_i assigns a value to a variable, p_i is referred to as a valuation. Correspondently, the information set, I, is referred to as a valuation system, V.

The third component is an aggregation function, f, which relates Y to the X_i.

$Y=f(X_1, ..., X_n)$

[*] Department of EECS, University of California, Berkeley, CA 94720-1776; Telephone: 510-642-4959; Fax: 510-642-1712; E-Mail: zadeh@eecs.berkeley.edu. Research supported in part by ONR N00014-02-1-0294, BT Grant CT1080028046, Omron Grant, Tekes Grant, Chevron Texaco Grant and the BISC Program of UC Berkeley.

The principal difference between CW and conventional systems of computation is that CW allows inclusion in the information set, I, of propositions expressed in a natural language, that is, linguistic valuations. Legalization of linguistic valuations has important implications. First, it greatly enhances the capability of computational methodologies to deal with imperfect information, that is, information which in one or more respects is imprecise, uncertain, incomplete, unreliable, vague or partially true. In realistic settings, such information is the norm rather than exception. Second, in cases in which there is a tolerance for imprecision, linguistic valuations serve to exploit the tolerance for imprecision through the use of words in place of numbers. And third, linguistic valuations are close to human reasoning and thus facilitate the design of systems which have a high level of machine intelligence, that is, high level of MIQ (machine IQ).

What does Computing with Words have to offer? The answer rests on two important tools which are provided by the machinery of fuzzy logic. The first tool is a formalism for mm-precisiation of propositions expressed in a natural language through representation of the meaning of a proposition as a generalized constraint of the form X isr R, where as noted earlier X is the constrained variable, R is the constraining relation and r is the modality of the constraint (Zadeh 1986).

The second tool is a formalism for computing with mm-precisiated propositions through propagation and counterpropagation of generalized constraints. The principal rule governing constraint propagation is the Extension Principle (Zadeh 1965, 1975). In combination, these two tools provide an effective formalism for computation with information described in a natural language. And it is these tools that serve as a basis for legalization of linguistic valuations.

What is important to note is that the machinery of fuzzy if-then rules—a machinery which is employed in almost all applications of fuzzy logic—is a part of the conceptual structure of CW.

Biographical Note

Professor Lotfi A. Zadeh

LOTFI A. ZADEH is a Professor in the Graduate School, Computer Science Division, Department of EECS, University of California, Berkeley. In addition, he is serving as the Director of BISC (Berkeley Initiative in Soft Computing).

Lotfi Zadeh is an alumnus of the University of Tehran, MIT and Columbia University. He held visiting appointments at the Institute for Advanced Study, Princeton, NJ; MIT, Cambridge, MA; IBM Research Laboratory, San Jose, CA; AI Center, SRI International, Menlo Park, CA; and the Center for the Study of Language and Information, Stanford University. His earlier work was concerned in the main with systems analysis, decision analysis and information systems. His current research is focused on fuzzy logic, computing with words and soft computing, which is a coalition of fuzzy logic, neurocomputing, evolutionary computing, probabilistic computing and parts of machine learning.

Lotfi Zadeh is a Fellow of the IEEE, AAAS, ACM, AAAI, and IFSA. He is a member of the National Academy of Engineering and a Foreign Member of the Russian Academy of Natural Sciences, the Finnish Academy of Sciences, the Polish Academy of Sciences, Korean Academy of Science & Technology and the Bulgarian Academy of Sciences. He is a recipient of the IEEE Education Medal, the IEEE Richard W. Hamming Medal, the IEEE Medal of Honor, the ASME Rufus Oldenburger Medal, the B. Bolzano Medal of the Czech Academy of Sciences, the Kampe de Feriet Medal, the AACC Richard E. Bellman Control Heritage Award, the Grigore Moisil Prize, the Honda Prize, the Okawa Prize, the AIM Information Science Award, the IEEE-SMC J. P. Wohl Career Achievement Award, the SOFT Scientific Contribution Memorial Award of the Japan Society for Fuzzy Theory, the IEEE Millennium Medal, the ACM 2001 Allen Newell Award, the Norbert Wiener Award of the IEEE Systems, Man and Cybernetics Society, Civitate Honoris Causa by Budapest Tech (BT) Polytechnical Institution, Budapest, Hungary, the V. Kaufmann Prize, International Association for Fuzzy-Set Management and Economy (SIGEF), the Nicolaus Copernicus Medal of the Polish Academy of Sciences, the J. Keith Brimacombe IPMM Award, the Silicon Valley Engineering Hall of Fame, the Heinz Nixdorf MuseumsForum Wall of Fame, other awards and twenty-six honorary doctorates. He has published extensively on a wide variety of subjects relating to the conception, design and analysis of information/intelligent systems, and is serving on the editorial boards of over sixty journals.

Professor in the Graduate School, Computer Science Division
Department of Electrical Engineering and Computer Sciences
University of California
Berkeley, CA 94720 -1776
Director, Berkeley Initiative in Soft Computing (BISC)

WORLDCOMP/GCA Keynote: Cloud Computing - The Next Revolution in Information Technology

Professor Dr. Rajkumar Buyya,
Director of CLOUDS Lab, The University of Melbourne, Australia
CEO, Manjrasoft Pvt Ltd, Melbourne, Australia
Recipient of the 2009 IEEE Medal for Excellence in Scalable Computing

Abstract:

Computing is being transformed to a model consisting of services that are commoditised and delivered in a manner similar to utilities such as water, electricity, gas, and telephony. In such a model, users access services based on their requirements without regard to where the services are hosted. Several computing paradigms have promised to deliver this utility computing vision and they include Grid computing, P2P computing, and more recently Cloud computing. The latter term denotes the infrastructure as a “Cloud” in which businesses and users are able to access applications from anywhere in the world on demand. Cloud computing delivers infrastructure, platform, and software (application) as services, which are made available as subscription-based services in a pay-as-you-go model to consumers. These services in industry are respectively referred to as Infrastructure as a Service (Iaas), Platform as a Service (PaaS), and Software as a Service (SaaS). To realize Cloud computing potential, vendors such as Amazon, Google, Microsoft, and IBM are starting to create and deploy Clouds in various locations around the world. In addition, companies with global operations require faster response time, and thus save time by distributing workload requests to multiple Clouds in various locations at the same time. This creates the need for establishing a computing atmosphere for dynamically interconnecting and provisioning Clouds from multiple domains within and across enterprises. There are many challenges involved in creating such Clouds and Cloud interconnections.

This keynote (1) presents the 21st century vision of computing and identifies various IT paradigms promising to deliver the vision of computing utilities; (2) defines the architecture for creating market-oriented Clouds and computing atmosphere by leveraging technologies such as VMs; (3) provides thoughts on market-based resource management strategies that encompass both customer-driven service management and computational risk management to sustain SLA-oriented resource allocation; (4) presents the work carried out as part of our new Cloud Computing initiative, called Cloudbus: (i) Aneka, a software system for providing PaaS within private or public Clouds and supporting market-oriented resource management, (ii) internetworking of Clouds for dynamic creation of federated computing environments for scaling of elastic applications, (iii) creation of 3rd party Cloud brokering services for content delivery network and e-Science applications and their deployment on capabilities of IaaS providers such as Amazon and Nirvanix along with Grid mashups, and (iv) CloudSim supporting modelling and simulation of Clouds for performance studies; and (5) concludes with the need for convergence of competing IT paradigms for delivering our 21st century vision along with pathways for future research.

About the speaker:

Dr. Rajkumar Buyya is Professor of Computer Science and Software Engineering; and Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Australia. He is also serving as the founding CEO of Manjrasoft Pty Ltd., a spin-off company of the University, commercialising its innovations in Grid and Cloud Computing. He has authored and published over 300 research papers and four text books. The books on emerging topics that Dr. Buyya edited include, High Performance Cluster Computing (Prentice Hall, USA, 1999), Content Delivery Networks (Springer, Germany, 2008) and Market-Oriented Grid and Utility Computing (Wiley, USA, 2009). He is one of the highly cited authors in computer science and software engineering worldwide (h-index=46, g-index=98, 11000+ citations).

Dr. Buyya has contributed to the creation of high-performance computing and communication system software for Indian PARAM supercomputers. He has pioneered Economic Paradigm for Service-Oriented Distributed Computing and developed key Grid and Cloud Computing technologies such as Gridbus and Aneka that power the emerging e-Science and e-Business applications. Software technologies for Grid and Cloud computing developed under Dr. Buyya's leadership have gained rapid acceptance and are in use at several academic institutions and commercial enterprises in 40 countries around the world.

Dr. Buyya has led the establishment and development of key community activities, including serving as foundation Chair of the IEEE Technical Committee on Scalable Computing and four IEEE conferences (CCGrid, Cluster, Grid, and e-Science). He has presented over 200 invited talks on his vision on IT Futures and advanced computing technologies at international conferences and institutions in Asia, Australia, Europe, North America, and South America. These contributions and international research leadership of Dr. Buyya are recognised through the award of "2009 IEEE Medal for Excellence in Scalable Computing" from the IEEE Computer Society, USA. For further information on Dr. Buyya, please visit his cyberhome: www.buyya.com.

SESSION

GRID SERVICES, SCHEDULING, AND RESOURCE MANAGEMENT + RELATED ISSUES

Chair(s)

TBA

Novel Mechanism for evaluating feedback in the Grid Environment on resource allocation

V.Vijayakumar[1], R.S.D.Wahida Banu[2], and J. H. Abawajy[3]

[1]PhD Research Scholar, Faculty of Information and Communication Engineering, Anna University, Chennai, Tamilnadu, India
[2]PhD Research Supervisor, Faculty of Information and Communication Engineering, Anna University, Chennai, Tamilnadu, India
[3]Pervasive Computing and Networks Research Director, Faculty of Science and Technology, Deakin University, Geelong, Victoria, Australia

Abstract

The primary concern in proffering an infrastructure for general purpose computational grids formation is security. Grid implementations have been devised to deal with the security concerns. The chief factors that can be problematic in the secured selection of grid resources are the wide range of selection and the high degree of strangeness. Moreover, the lack of a higher degree of confidence relationship is likely to prevent efficient resource allocation and utilization. In this paper, we propose an efficient approach for the secured selection of grid resources, so as to achieve secure execution of the jobs. The presented approach utilizes trust and reputation for securely selecting the grid resources by also evaluating user's feedback on the basis of the feedback already available about the entities. The proposed approach is scalable for an increased number of resources.

Index Terms: *Security, Resource Selection, Trust, Self-Protection Capability, Reputation, Feedback Evaluation.*

1. Introduction

The latest developments in wide-area network performance have facilitated the emergence of grid computing as a practical archetype to satisfy the growing demand for computation power that could not be fulfilled with the aid of the internal resources of a single organization [14]. The objective to distribute processing resources amidst several organizations in order to resolve large scale problems has resulted in the introduction of computational grids [1, 2]. Grids have developed into widespread platforms for high-performance and resource-intensive applications due to the reason that they encompass vast potential of capabilities that can assist large distributed applications [15].

The apprehensive sharing is not primarily file exchange, rather direct access to computers, software, data and other resources that are essential for diverse collaborative problem-solving and resource-brokering strategies emerging in industry, science and engineering [31]. A range of phenomena including (a) geographical allocation of resources, (b) resource heterogeneity, (c) autonomously managed grid domains comprising of their own resource policies and practices, and (d) Grid domains employing diverse access and cost models, generate huge challenges for resource management in grid systems.

Resource and security assurance are the two fundamental requirements of Grid applications [29], [30]. The solid problems underneath the grid concept include coordinated resource sharing and problem resolving in dynamic, multi-institutional virtual organizations [19]. Infected grid resources can probably wreck the applications running on the same grid platform via the malicious codes designed by intruders

At present, security is integrated in the grid toolkits (e.g. the Globus toolkit [21]) employed at the provider sites (parties that provide resources for use in the grid). Secure channels, authentication [32], unsupervised login, delegation, and resource usage [31] are all administered by the toolkit. However, these mechanisms do not offer security of the grid user (the person or entity who desires to utilize resources). The user is forced to trust the provider without confirming the justification of the trust [20]. Users submit jobs to far-off resources and generally have no clear authority over the resources themselves. Therefore, mutually the users and resources can be considered as independent agents, possessing control of their own behavior. The independence causes an increase in intrinsic insecurity owing to the fact that an individual is not capable of predicting the response of another to varying situations. It is essential that the grid service providers proffer guaranteed security, privacy protection, and dependable accessibility of all Grid-enabling platforms [30].

This paper discusses a novel approach for selecting grid resources on the basis of trust and reputation, to execute the jobs in a secured manner. Our earlier researches [12], [13] and [16] have been extended with a novel and effective approach presented for evaluating user's feedback. The chief objective of this research is to develop an approach that is capable of availing trust and reputation based security for Grid resource selection towards scheduling large number of independent and indivisible jobs. The proposed approach performs the scheduling of the incoming jobs in accordance with the computed Trust Factor value. The feedback from user communities and the feedback received from other entities in the Grid are employed in determining an entity's reputation weightage which in turn is utilized along with Self-protection capability to compute the entity's Trust Factor (TF) value. The ability of an entity to handle intrusions, viruses, unauthorized access and secured file storage are denoted as self protection capability of that site. The reputation mechanisms provide a way for building trust through social control using community based feedback about past experiences of entities. Besides, a novel approach is proposed for evaluating user's feedback. The feedback given by a user about an entity is evaluated on the basis of the aggregated feedback available about that entity.

The rest of the paper is organized as follows: In Section 2, a concise review of the recent researches related to the proposed approach is presented. An overview of Trust and Reputation in the context of Grid computing is provided in Section 3. The proposed efficient approach for secured Grid resource selection and the devised novel approach for evaluating user's feedback are explained in Section 4 and Section 5 concludes the paper.

2. Review of related Researches

A brief review of some of the literature dealing with trust management and reputation-based security mechanisms for improving the performance of grid computing are given below which serve as the motivation behind the proposed work.

A formal definition of both trust and reputation was presented and a model for incorporating trust into Grid systems was discussed by Farag Azzedin and Muthucumaru Maheswaran [3]. An overview of an open source Grid toolkit known as Grid bus, the architecture of which is fundamentally driven by the requirements of Grid economy was carried out by Rajkumar Buyya and Srikumar Venugopal [4]. Grid bus technologies proffer services for both computational and data grids that control the budding eScience and eBusiness applications. A self-regulating system for P2P network that works on robust reputation mechanism was proposed by Ernesto Damiani et al. [10]. They realized reputation sharing with the aid of the distributed polling algorithm. A general-purpose resource selection framework was put forth by Chuang Liu et al. [11] they have defined a resource selection service for positioning Grid resources that conform to application requirements and evaluated them on basis of specified performance model and mapping strategies, and returned an appropriate collection of resources, if any were present.

Abawajy et al. [17] have proposed an approach that makes use of credit score and reputation score to manage creditworthiness and trustworthiness of the Grid Service Customer (GSC) and Grid Service Provider (GSP) respectively. A Bayesian network-based trust model and a method for building reputation based on recommendations in peer-to-peer networks was put forth by Yao Wang and Julita Vassileva [9]. Sepandar D. Kamvar et al. [23] proposed a reputation management system known as Eigen Trust, which was capable of effectively decreasing the number of downloads of inauthentic files in a P2P system. The reputation value of every peer is calculated with the aid of the number of successful downloads and the "opinions" of other peers.

A novel fuzzy-logic trust model for securing Grid computing across multiple resources sites was devised by Shanshan Song and Kai Hwang [5]. They built a novel Grid security scheme known as SARAH supported by encrypted channels amid private networks. Justin R.D. Dyson et al. [22] illustrated a trust framework model for Grid computing that facilitates users to implement their jobs on reliable and efficient resources, thus fulfilling clients' quality-of-service (QoS) requirements.

A reputation-based trust supporting framework that encompasses a coherent adaptive trust model for quantifying and comparing the trustworthiness of peers on basis of a transaction-based feedback system and a decentralized implementation of such a model over a structured P2P network was illustrated by Li Xiong and Ling Liu [24]. A brief review of the contemporary state of Globus was carried out by Ian Foster [21]. The spotlight was on those features of the GT4 release that should sound interesting to those aspiring to work with the software. A trust brokering system which operates in a peer-to-peer fashion was proposed by Farag Azzedin and Muthucumaru Maheswaran [25]. They built a security-aware model involving resource providers and the consumers that segregates the concepts of accuracy and honesty. A novel fuzzy-logic trust model for securing Grid resources was put forth by Shanshan Song et al. [26].

They have as well built a SeGO scheduler that aids in trusted Grid resource allocation.

Chunqi Tian et al. [27] presented the ARTrust—an Attack Resistant Trust management model which is a novel recommendation on basis of trust model for P2P networks. A trust model used to compute and compare the trustworthiness of entities in the same autonomous and different domains was proposed by Baolin Ma et al. [28]. This model proffers various methods to handle the problems of users and related resources belonging to the identical or diverse domains. Nadia Ranaldo and Eugenio Zimeo [18] put forth a framework for brokering of Grid resources, virtualized through web Services where in a dynamically configuration is possible with respect to multiple syntactic and semantic description languages and related matching strategies. Zhiguo Shi et al. [19] described a novel anonymous coordination authentication scenario capable of providing efficient and reliable anonymous identity authentication and remote platform attestation for Grid computing systems. Lohr et al. [20] illustrated an approach to improve the Grid security with the aid of a combination of trusted computing and virtualization technologies.

3. Trust and Reputation

This section comprehends a prologue of trust and reputation in the context of Grid Computing.

3.1 Trust

Trust is not a black and white substance. Often, there exists a grey area in conveying the trustworthiness of a computer site [5]. Like the human relationships, trust is as well denoted by a linguistics term rather numerically. The definition of trust as given by Farag Azzedin and Muthucumaru Maheswaran [3] is as follows:

Trust is the firm belief in the competence of an entity to act as expected, such that, the firm belief is not a fixed value associated with the entity but rather it is subject to entity's behavior and applies only within a specific context at a given time.

The firm belief can be defined as a dynamic value which is found to span over a set of values varying from very trustworthy to very untrustworthy. The trust factor is developed on the basis of earlier experiences and is provided for a particular context. The trust factor is associated with a given time instance, as the trust level involving two entities is not essentially the same for today when compared to a year ago.

3.2 Reputation

Reputation is defined as a measure of trustworthiness in the sense of reliability. Reputation systems [6] proffer a scheme for creating trust through social control devoid of trusting third parties. By means of community based feedback about past experiences of entities; reputation mechanisms offer a scheme for building trust through social control. This aids in arriving at suggestions and judgment on quality and consistency of the transactions [7]. According to Farag Azzedin and Muthucumaru Maheswaran [3], reputation of an entity is defined as an expectation of its behavior based on other entities' observations or information about the entity's past behavior at a given time.

4. Secured Resource Selection for Scheduling Jobs

This section fine points the proposed approach for resource selection deliberated for secure scheduling of independent and individual jobs to grid sites. The wide range of resources and the strangeness of entities serve as difficulties during the process of resource selection. Owing to the fact that a high-proficient society is not capable of getting along with a high-trustworthy social relationship, it is impossible to attain efficient resource sharing in Grid without certain trust relationship core. It is possible for entities to rely on other entities for the information regarding a particular entity while arriving at trust based decisions. This can be achieved by a reputation mechanism. With due concern over the aforesaid conditions, we have proposed an approach by incorporating both trust and reputation.

The proposed approach aims for secure scheduling of incoming jobs to the available resource sites based on the Trust Factor value. The Trust Factor (TF) value of each resource site is determined with the aid of its self-protection capability and reputation weightage acquired from user community regarding its past behavior. There are two essential postulations that are made: (a) all resource sites have prior agreements to participate in the Grid operations; and (b) the Grid sites truthfully report their self- protection capability to Grid organization manager (GOM). Selfish Grids [8] have not been dealt with, in the proposed scheme.

4.1 Self-Protection capability

The self-protection capability of the entities in a grid organization is determined by the grid organization manager. Now and then, every entity reports its self-protection capability trustfully and truthfully to the GOM.

An aggregation of the values of the security factors given below is employed in determining the self-protection capability of an entity. The value of these factors differs in the range between 0 and 1.

Based on the security, a weightage is allocated to all the security factors and at the final point is aggregated to calculate the self-protection capability. Table 1 provides the weightage allocated to security factors.

TABLE 1. Weightage of Security Factors

Security Factors	Weightage (W)
IDS Capabilities	0.825
Anti-virus Capabilities	0.85
Firewall Capabilities	0.9
Authentication Mechanism	0.8
Secured File Storage Capabilities	0.7
Interoperability	0.6
Secured Job Execution	0.75
Authorization Mechanism	0.87

The self-protection capability is calculated using the following formula

$$SPC = \sum_{i=1}^{n} W(i) * A(i) \qquad (1)$$

where, n is the total number of factors, W is the weightage and A (i) is the value of the factor.

4.2 Reputation Computation

Given that, reputation is a versatile concept [9], there are numerous aspects in it, for instance truthfulness, honesty and the like. The reputation weightage of an entity is determined on the basis of the feedback provided by both the user community and other entities in the grid, regarding the entity's security characteristics and their previous experiences. On completion of a job, the user will offer feedback on the attributes to the Reputation manager (RM) according to their experience. Similarly, the entities in the grid provide feedback to the RM on a timely basis. The value of feedback lies in the range between 0 and 1. The feedback given by the users and the entities are aggregated and employed in the proposed approach. In addition, an effective approach for assessing the feedback given by the user is presented. The reputation weightage is determined by using the algorithm given in section 4.2.1. The RM in grid organization contains the reputation weightage of every entity.

4.2.1 Algorithm for Reputation weightage calculation

The aggregated feedback of all the security attributes of an entity is denoted in the form of a Reputation Vector (RV),

$$R_v = [SA_1, SA_2, ..., SA_n]$$

where n is the total number of security attributes.

The aggregated feedback of all the entities in the Grid domain is represented as a Reputation Matrix (RM) given as follows. Every row in RM denotes the reputation vector (RV) of an entity.

$$R_M = \begin{bmatrix} SA_{11} & \mathrm{SA}_{12} & \mathrm{SA}_{13} .... SA_{1j} \\ SA_{21} & \mathrm{SA}_{22} & \mathrm{SA}_{23} ...... SA_{2j} \\ SA_{i1} & \mathrm{SA}_{i2} & \mathrm{SA}_{i3} ...... SA_{ij} \end{bmatrix}$$

where i denote the number of entities and j denotes the number of attributes.

The reputation weightage of each entity is determined by considering its relativity with other entities in the Grid domain through the construction of a relativity matrix. The relativity matrix is formed as follows.

$$\mathrm{Re}l_{Mat} = \begin{bmatrix} \varphi(\mathrm{E}_1, E_1) & \varphi(\mathrm{E}_1, E_2) & \varphi(\mathrm{E}_1, E_3) \cdots\cdots \varphi(\mathrm{E}_1, E_i) \\ \varphi(E_2, E_1) & \varphi(\mathrm{E}_2, E_2) & \varphi(\mathrm{E}_2, E_3) \cdots\cdots \varphi(E_2, E_i) \\ \varphi(E_3, E_1) & \varphi(\mathrm{E}_3, E_2) & \varphi(\mathrm{E}_3, E_3) \cdots\cdots \varphi(E_3, E_i) \\ \vdots & \vdots & \vdots \qquad\qquad \vdots \\ \varphi(\mathrm{E}_i, E_1) & \varphi(\mathrm{E}_i, E_2) & \varphi(\mathrm{E}_i, E_3) \cdots\cdots \varphi(E_i, E_i) \end{bmatrix}$$

where i is the number of entities and $\varphi(E_a, E_b)$ denotes the relativity between the entities E_a and E_b. The relativity between two entities E_a and E_b is determined by correlating the security attributes of that entities. Equation (2) is used to determine the relativity between the securities attributes of the two entities:

$$\varphi_1(A_m, A_n) = \begin{cases} 1 \ , & A_m > A_n \\ 0 \ , & A_m < A_n \\ 0.5, & A_m = A_n \end{cases} \qquad (2)$$

Subsequently, the relativity between the entities $\varphi(E_a, E_b)$ is computed using the following equation (3).

$$\varphi(E_a, E_b) = \frac{\sum_{f=1}^{j} \varphi_1\left(A_{f(E_a)}, A_{f(E_b)}\right)}{j} \tag{3}$$

Ultimately the reputation weightage is determined with the aid of the following equation (4).

$$RW(E_a) = \sum_{b=1}^{n} \varphi(E_a, E_b) \tag{4}$$

A greater value of RW signifies that the entity has a better reputation and a smaller value denotes that the entity has a minimal reputation.

4.3. Trust Factor calculation and Resource selection

The trust factor (TF) of every entity is determined by means of the self – protection Capability (SPC) and Reputation Weigtage (RW) computed in the above sections, by employing the following equation.

$$TF(E_a) = SPC(E_a) + RW(E_a) \tag{5}$$

The following algorithm is employed in the selection of the resource for the secured execution of incoming jobs.

Initially, the Trust Factor (TF) is computed for all the entities in the grid. Then, the calculated TF values are sorted in descending order along with the index. Finally, based on the sorted index values, the jobs are allocated to the respective entities.

```
for  h = 1 to i
        Obtain SPC from GOM
        Obtain RW from RM
        Calculate TF
end

[STF, Ind] = DescSort(TF)

for  k = 0 to N
        Allocate Entity [Ind[k]] to job $J_k$
end
```

where N is the total number of jobs.

4.4 Novel approach for User's feedback evaluation

Once the job ends, the users will be impelled to supply a feedback about the entity that consummates their job. If the user is not a trusted person, the feedback given by the user may be wrong. Sometimes, by fault, the users may enter immoral values. The above factors affect the calculation of the entities' reputation weightage and are likely to result in wrong selection of resources for job execution. This has necessitated the need for evaluating user's feedback before taking it into consideration.

The feedback values of an entity given by the users are evaluated based on the aggregated feedback available for that entity. If the current feedback given by the user deviates from the existing feedback, the current feedback values are not taken into consideration. Moreover, the users are prompted to check the feedback values given by them. The steps in the novel approach proposed for evaluating the user's feedback are described below.

The existing feedback values and the current feedback provided by the user are denoted as follows:

Existing feedback, $E_F = [E_{f1} \; E_{f2} \; E_{f3} \; \cdots \; E_{fn}]$

Current feedback, $C_F = [C_{f1} \; C_{f2} \; C_{f3} \; \cdots \; C_{fn}]$

where n = 10

Initially, the mean $\overline{E_F}$ of the existing feedback values (E_F) is computed using the following equation (6).

$$\overline{E_F} = \sum_{i=1}^{n} \frac{E_{F(i)}}{n} \tag{6}$$

The individual values in the current feedback (C_F) are subtracted from the mean value ($\overline{E_F}$) to obtain EC_F. The above process is stated in equation (7).

$$[EC_F] << abs(\overline{E_F} - C_{F(i)}) \tag{7}$$

The deviated values in the current feedback C_F, provided by the user, are identified by the following steps. Initially, a correlation matrix C_M is formed for EC_F as follows:

$$C_M = \begin{bmatrix} d_{00} & d_{01} & d_{02} & \cdots & d_{0j} \\ d_{10} & d_{11} & d_{12} & \cdots & d_{1j} \\ d_{20} & d_{21} & d_{22} & \cdots & d_{2j} \\ d_{30} & d_{31} & d_{32} & \cdots & d_{3j} \\ \vdots & & & & \\ d_{i0} & d_{i1} & d_{i2} & \cdots & d_{ij} \end{bmatrix}$$

where $d_{ij} = abs\left(EC_{F(i)} - EC_{F(j)}\right)$

Afterwards, the value pairs in the correlation matrix C_M, for which the difference is greater than a threshold value are selected and represented as S_p. The threshold value is computed by multiplying the mean value $\overline{EC_F}$ with 2. The aforesaid procedure is formulated in equations (8) and (9).

$$Thresh = \overline{EC_F} \times 2 \tag{8}$$

$$S_p = \{(x, y) : p(z)\} \tag{9}$$

where $p(z) = C_{M_{ij}} > Thresh$

Finally, the frequent value in the formed set Sp is chosen and the feedback value corresponding to it is identified as the deviated value. If any deviated value is identified, the users are prompted to check the values given by them. On the basis of the user's response, the feedback is either considered or discarded.

5. Conclusion

The proficient utilization of Grid computing facilities vitally necessitates highly sophisticated and secured resource management systems. Moreover, accessing and sharing of resources necessitate the assurance of high trustworthiness as an inevitable factor. Reputation mechanisms provide a way for creating trust through social control with the assistance of feedback concerning the past experiences. In this paper, we have proposed an efficient approach for the selection of the appropriate resource for the secured execution of the job. Security for the resource selection procedure is offered by the proposed approach by combining trust and reputation. We have presented a novel approach for evaluating the feedback provided by the users. The approach proposed has been proved to be efficient in choosing a secured entity from a pool of available ones.

References

[1]F.Berman,G. Fox and T. Hey (eds.), Grid Computing: Making the Global Infrastructure a Reality. Wiley, 2003.

[2] M. Cosnard and A. Merzky, "Meta- and Grid-Computing", in Proceedings of the 8th International Euro-Par Conference, August 2002, pp. 861–862.

[3] Farag Azzedin, Muthucumaru Maheswaran, "Towards Trust-Aware Resource Management in Grid Computing Systems," ccgrid, p. 452, 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'02), 2002.

[4] R. Buyya and S. Venugopal, The Grid bus Toolkit for Service Oriented Grid and Utility Computing: An Overview and Status Report, Proceedings of the First IEEE International Workshop on Grid Economics and Business Models (GECON), 2004.

[5] Shanshan Song and Kai Hwang, Dynamic Grid Security with Trust Integration and Optimized Resource Allocation, Internet and Grid Computing Laboratory, University of Southern California, Los Angeles, CA. 90089 USA.

[6] R. A. Malaga. Web-based reputation management systems: Problems and suggested solutions. Electronic Commerce Research, 1(4), 2001.

[7] P. Resnick, R. Zeckhauser, E. Friedman, and K. Kuwabara. Reputation Systems. Communications of the ACM, 43(12), December 2000: 45–48.

[8] Y.-K. Kwok, S. Song and K. Hwang, "Selfish Grid Computing: Game-Theoretic Modeling and NAS Performance Results", in Proceedings of CCGrid 2005, Cardiff, UK, May 2005.

[9] Yao Wang and Julita Vassileva: Trust and Reputation Model in Peer-to-Peer Networks. In Proceedings of the 3rd IEEE International Conference on Peer-to-Peer Computing. Linkŏping: IEEE Computer Society (2003), 150–158.

[10] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, P. Samarati and F. Violante, "A Reputation-Based Approach for Choosing Reliable Resources in Peer-to-Peer Networks", in Proceedings of ACM CCS 2002.

[11] C. Liu, L. Yang, I. Foster and D. Angulo, "Design and Evaluation of a Resource Selection Framework for Grid Applications", in Proceedings of HPDC-11, 2002.

[12] Vijayakumar, V. And Wahidha Banu, R. S. D., "Trust and Reputation Aware Security for Resource Selection in Grid Computing," International Conference on Security Technology (SECTECH '08), pp: 121-124, Dec 13-15, 2008.

[13] V.Vijayakumar and R.S.D.Wahida Banu, "Security for Resource Selection in Grid Computing Based On Trust and Reputation Responsiveness", International Journal of Computer Science and Network Security (IJCSNS), Vol.8, No.11, pp. 107-115, November 2008.

[14] Foster, I., Kesselman, C., Nick, J., Tuecke, and S.: The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Technical Report, Open Grid Service Infrastructure WG, Global Grid Forum, 2002.
[15] Pautasso, C., Alonso, G.: Parallel Computing Patterns for Grid Workflows. In: the HPDC2006 Workshop on Workflows in Support of Large-Scale Science. France, 2006.
[16] V.Vijayakumar and R.S.D.Wahida Banu, "Secured Resource Selection in Grid Computing: Trust and Reputation Sentient Scheme", Communications in Computer and Information Science (CCIS), Vol. 27, Springer, 2009.
[17] J. H. Abawajy and A. M. Goscinski, "A Reputation-Based Grid Information Service ," Lecture Notes in Computer Science (LNCS), Springer, Vol. 3994/2006, pp. 1015-1022, 2006.
[18] Nadia Ranaldo, Eugenio Zimeo. A Framework for QoS-based Resource Brokering in Grid Computing. In 5th IEEE ECOWS, the 2nd Workshop on Emerging Web Services Technology, Halle, Germany, 2007.
[19] Zhiguo Shi, Yeping He, Xiaoyong Huai, Hong Zhang. Identity Anonymity for Grid Computing Coordination based on Trusted Computing. Proceedings of the Sixth International
Conference on Grid and Cooperative Computing. pp.403-410, 2007.
[20] Lohr, H. Ramasamy, H. V. Sadeghi, A.-R. Schulz, S. Schunter, M. Stuble, C., Enhancing Grid Security Using Trusted Virtualization, Lecture Notes in Computer Science, pp. 372-384, Springer, 2007.
[21] I. Foster. Globus toolkit version 4: Software for service-oriented systems. In Proc. of the IFIP International Conference on Network and Parallel Computing, 2005.
[22] J. R. D. Dyson, N. Griffiths, H. N. Lim Choi Jeung, S. A. Jarvis, and G. R. Nudd, Trusting Agents for Grid Computing, in Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC 2004), pp. 3187-3192, IEEE Press, October 2004.
[23] S.D. Kamvar, M.T. Schlosser and H. Garcia-Molina, “The Eigentrust Algorithm for Reputation Management in P2P Networks”, in Proceedings of ACM WWW 2003.
[24] L. Xiong and L. Liu, “PeerTrust: Supporting Reputation-based Trust to P2P E-Communities”, IEEE Trans. Knowledge and Data Engineering, July 2004, pp. 843–857.
[25] F. Azzedin and M. Maheswaran, “A Trust Brokering System and Its Application to Resource Management in Public- Resource Grids”, in Proceedings of IPDPS 2004.
[26] S. Song, K. Hwang and M. Macwan, “Fuzzy Trust Integration for Security Enforcement in Grid Computing”, in Proceedings of IFIP International Conf. on Network and Parallel Computing, (NPC-2004), Wuhan, China, October 18–20, 2004, pp. 9–21.
[27] Chunqi Tian, Shihong Zou, Wendong Wang, Shiduan Cheng, An Efficient Attack-Resistant Trust Model for P2P Networks, IJCSNS, Vol. 6 No. 11 pp. 251-258, 2006.
[28] Baolin Ma, Jizhou Sun, Ce Yu, Reputation-based Trust Model in Grid Security System, Journal of Communication and Computer, Volume 3, No.8 (Serial No.21), 2006.
[29] F. Berman, R. Wolski, H. Casanova, W. Cirne, H. Dail, M. Faerman, S. Figueira, J. Hayes, G. Obertelli, J. Schopf, G. Shao, S. Smallen, N. Spring, A. Su and D. Zagorodnov, “Adaptive Computing on the Grid Using AppLeS”, IEEE Trans. on Parallel and Distributed Systems, Vol. 14, April 2003.
[30] V.Welch, F. Siebenlist, I. Foster, J. Bresnahan, K. Czajkowski, J. Gawor, C. Kesselman, S.Meder, L. Pearlman and S. Tuecke, “Security for Grid Services”, in Proceedings of the HPDC-12, 2003.
[31] Foster, I., Kesselman, C., Tsudik, G. and Tuecke, S. A Security Architecture for Computational Grids. ACM Conference on Computers and Security, 1998, pp: 83-91.
[32] J. Basney, W. Nejdl, D. Olmedilla, V. Welch, and M. Winslett. Negotiating trust on the grid. In 2nd Workshop on Semantics in P2P and Grid Computing, New York, May 2004.

A Min-Min-based Job Scheduling Algorithm for Fault-Tolerant Large Scale Computational Grid Systems

Chao-Chin Wu, Lien-Fu Lai, and Jia-Xian Lai
Department of Computer Science and Information Engineering
National Changhua University of Education, Changhua City, Taiwan, R.O.C.

Abstract - *Previously, we proposed a security-aware genetic algorithm for job scheduling to address the problem of the heterogeneity of fault-tolerance mechanisms in a computational grid, where the risk nature of the grid environment is also taken into account. This paper proposes a new job scheduling algorithm having the following features. The algorithm is Min-Min based rather than genetic algorithm. The fault-tolerance mechanisms considered are the retry, the migration without checkpointing, the migration with checkpointing and the replication mechanisms. In the new algorithm, each computational node supports all four mechanisms. The access to the storage node for data input is modeled in the new algorithm, which is ignored in the previous algorithm. Simulation results show that the security-aware Min-Min-based algorithm outperforms the original Min-Min and the sufferage algorithms. The proposed algorithm can reduce not only the makespan but also the failures.*

Keywords: Min-Min, job scheduling, security aware, fault tolerance, grid system.

1 Introduction

A computational grid is a hardware and software infrastructure that pro-vides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities [1]. However, a large-scale grid system is inherently unreliable by nature. The jobs are subject to system failures or delays caused by infected hardware, software vulnerability, network failure, overloaded resource conditions, non-availability of required software components, and distrusted security policy. To address these problems, a variety of fault-tolerance mechanisms and supports have been proposed for Grid systems [2-5].

Despite the fact that many heuristics have been suggested for large-scale job scheduling [6-10], as pointed out by Song et al. [11], they are not applicable in a risky environment. Therefore, Song et al. developed security assurance and risk-resilient strategies and proposed eight job-scheduling algorithms for use under various risky conditions to address these problems. Their proposed security-assured job scheduling strategies consider the risk relationship between jobs and nodes using the security demand (SD) and the trust level (TL).

Although Song et al. proposed eight job-scheduling algorithms for use under various risky conditions [11, 12], all of these algorithms ignore the heterogeneity of fault-tolerance mechanisms supported in grid systems, where the scheduler assumes that all computational nodes adopt the same fault-tolerance strategy. In fact, different computational nodes are usually protected by different fault-tolerance mechanisms because distributed computational nodes are managed by different autonomous domains in a realistic large-scale computational grid. In such a grid environment, all the job scheduling algorithms proposed by Song et al. are not applicable. Therefore, previously we proposed a security-assured grid job-scheduling strategy considering the heterogeneity of fault-tolerance mechanism support [13]. In this algorithm, we considered four kinds of fault-tolerance mechanisms, including the job retry (JRT), the job migration without checkpointing (JMG), the job migration with check-pointing (JCP) and the job replication (JRP) mechanisms. The scheduler will decide which kinds of fault-tolerance mechanisms will be applied to each individual job for more reliable computation and shorter makespan. To encode mixed kinds of fault-tolerance mechanisms into a single chromosome, we also proposed a new chromosome encoding approach. The proposed genetic algorithm has shorter makespan and provides more excellent efficiencies on improving the job failure rate than Min-Min and Sufferage algorithms [14].

In this paper, we consider a different grid model and propose a Min-Min based algorithm for job scheduling in a fault-tolerance large-scale computational grid. The new algorithm is Min-Min based, resulting in a lower computation complexity than that for the genetic algorithm. The previously proposed genetic algorithm adopts a Space-Time method to accelerate the convergence speed. The method relies on the scheduling history of each job to convergent the scheduling process efficiently. The proposed genetic algorithm may require longer scheduling time or have poor solutions when there are too many jobs that are submitted for the first time in a system. To address the problem, the new algorithm is not revolutionary approach. Instead, it is Min-Min based. For the system model, we assume that each computational node supports all the four fault-tolerance mechanisms. Each job will be protected by at least one of JRT, JMG and JCP. Moreover, JRP can be also applied for a job to provide a more reliable computation. When scheduling one job, in addition to the failure probability determined by the job security demand and the node trust level, we also consider the additional execution time required for the overhead of the applied fault-tolerance mechanism. The newly proposed algorithm aims to shorten the makespan of executing all the jobs by assigning a suitable computational node and an appropriate fault-tolerance algorithm for each job. Furthermore, the access time for reading data from storage nodes is also taken into account in the system model, where the contentions for network bandwidth and storage node bandwidth are modeled. Simulation results demonstrate that the proposed Min-Min algorithm can shorten the makespan and reduce the failure rate.

The rest of this paper is organized as follows. Section 2 introduces the related work. Section 3 presents the system model and the proposed job scheduling algorithm. Section 4 demonstrates the experimental results. Finally, Section 5 concludes the paper.

2 Related work

A variety of job scheduling strategies have been proposed for computational grids. Xiao et al. proposed an incentive-based scheduling scheme that investigates how to maximize the success rate of job execution and minimize the fairness deviation among resources by allowing resource providers and resource consumers to make autonomous scheduling decisions [15]. Doulamis et al. proposed a fair scheduling algorithm that will fairly reduce the CPU rates assigned to the tasks when the resources are insufficient so that the share of resources that each user gets is proportional to the user's weight [6]. Viswanathan et al. proposed a resource-aware dynamic incremental scheduling that handles large volumes of computationally intensive, arbitrarily divisible loads submitted for processing at cluster or grid systems [7]. Lee and Zomaya proposed two algorithms for bag-of-tasks applications, one for data-intensive tasks and one for computation-intensive tasks, which adopt task duplication for scheduling without requiring accurate performance prediction information [8]. Bertin et al. proposed a fully decentralized algorithm that can achieves both optimal path selection and flow control only requiring local information at each slave computing task and at each bu.er in the network links [9]. Nobrega et al. proposed scheduling heuristics that investigate the tradeoff between two factors: the challenging requirement of complete and accurate information about the applications and the grid environment, and the extra consumption of resources incurred by task replication [10]. Li et al. proposed a predictable and grouped genetic algorithm where a job workload estimation algorithm is designed to evaluate a job workload based on its historical execution records and the divisible load theory is employed to predict an optimal fitness value by which the convergence process can be shortened in searching for a large scheduling space [16]. Chang et al. proposed an ant-based algorithm that aims at balancing the entire system load while trying to minimize the makespan of a given set of jobs [17]. In addition, Chang et al. also proposed a job scheduling algorithm that dispatches a job to the site where the needed data are present to reduce data access time and the amount of inter-cluster-communications [18].

Some scheduling strategies especially emphasized the importance of security awareness. Azzedin and Maheswaran [19] proposed a trust model that incorporates the security implications into scheduling algorithms. Humphrey and Thompson [20] proposed usage models for security-aware Grid computing but they did not elaborate on how to design a scheduler by incorporating the security concerns into collaborative computing over distributed cluster environment. Abawajy [21] suggested replicating jobs at multiple sites to guarantee successful job executions in a Grid environment.

Song et al. [12] thought that a job distributed to a remote node may suffer from some infections or malicious attacks. Therefore, a job scheduler must consider the risk when dispatching jobs to remote nodes [12]. They proposed a job failure model to represent the risk level in job scheduling. There were two parameters in this model: the security demand (SD) and trust level (TL). SD represents a job's security requirement level, higher SD value represents this job has higher security requirement, so the job needs a more reliable node for job execution. TL represents a node's secure level, higher TL value represents this node can provide a more reliable environment. So different jobs assigned to different nodes would have different risk levels. Base on the job failure model, they proposed three schedule strategies based on different risk level: (1) Secure mode: jobs were only scheduled to those nodes which can ensure security. (2) Risky mode: jobs were scheduled to any available nodes without considering the risks between jobs and nodes, so it took all possible risks. (3) F-risky mode: jobs were scheduled to available nodes to take at most f risk, where f was a probability.

Moreover, in [11], Song et al. proposed four types of scheduling strategies: (1) Risky mode: jobs were scheduled to any available nodes without considering risks between jobs and nodes, so it took all possible risks. (2) Preemptive mode - failed jobs would be moved to another node, and then restarted from the beginning. (3) Replicated mode: in order to accomplish jobs safely, a job would be duplicated to multiple nodes for better success rate of job execution. (4) Delay-Tolerant mode: we would wait for a period of time to allow the nodes to have more time to deal with job execution. The simulation results mentioned that most fault tolerance algorithms are better than those algorithms without fault tolerance techniques in makespan, except Replication. The Replication approach produced too much workload for the node, so it had low makespan performance.

In [22], Chtepen et al. proposed several heuristics to enhance the efficiencies of two fault-tolerant techniques: job checkpointing and job replication. They dynamically adapted the checkpointing frequency and the number of replicas to strong variations in grid availability based on monitored grid state, job characteristics, and collected historical information. Moreover, in order to combine the advantages of both techniques, they also proposed a hybrid scheduling strategy that switches at runtime between checkpointing and replication depending on the system load. In [23], Hwang and Kesselman proposed a failure detection service (FDS) and a flexible failure handling framework (Grid-WFS) as a fault tolerance mechanism on the Grid. The FDS enables the detection of both task crashes and user-de.ned exceptions without requiring any modification to both the Grid protocol and the local policy of each Grid node. The Grid-WFS, built on top of FDS, uses workflow structure as a high-level recovery policy specification enables support for multiple failure recovery techniques, including the retrying on another available Grid resource strategy, the retrying on the same resource strategy and the restarting with checkpointing on the same resource strategy.

3 Proposed algorithm

The grid system considered in the work consists of geographically dispersed computational sites having different administrative polices and heterogeneous resources. Any computational node supports the following fault tolerance mechanisms for more reliable computation.

- Job retry (JRT) mechanism. The JRT mechanism is the simplest fault tolerance technique, which will re-execute the failed job from the beginning on the same computational node.
- Job migration without checkpointing (JMG) mechanism. The JMG mechanism will move the failed job to another computational node and re-execute the job from the beginning on the latter computational node.
- Job migration with checkpointing (JCP) mechanism. The JCP mechanism will record the state of the job periodically at rum time. If the job fails, it is moved to another computational node and resumed the execution from the last checkpoint.
- Job Replication (JRP) mechanism. The JRP mechanism replicates a job to multiple computational nodes such that the job has higher success rate. If one of those replicas has already completed, then all other replicas should stop their execution to save the computing power.

The job failure model adopted is similar to [11, 12, 13]. In this model, the risks between jobs and nodes are considered. The SD_i represents the security demand for the job i. The higher the SD_i value, the higher the security requirement for the job. TL_j represents the security guarantee for the node j, the higher the TL_j value, the higher the node reliability. The job failure probability is an exponential distribution as follows.

$$NP_j^i = \begin{cases} 0 & if\ SD_i \le TL_j, \\ 1 - e^{-\lambda(SD_i - TL_j)} & if\ SD_i > TL_j. \end{cases}$$

Each job has to access to one input file for its execution and each input file is replicated in more than one storage node. Whether a job can successfully access to its required input file or not depends on two parameters: connection failure and network contention. Each link from a computational node to a storage node is associated with a connection failure probability. Each storage node and each link has its own maximum transmission bandwidth. Computation nodes compete for storage node and network bandwidths before they can access to the required data for the assigned jobs.

The batch scheduling scheme was adopted in our model. In realistic Grid environment, users would transmit their jobs in anytime, so different jobs would have different arrival times. We set a scheduling cycle which has a fixed interval time, when a scheduling cycle begins we will schedule all jobs which were arriving in this scheduling cycle by one of our proposed job scheduling algorithms.

We propose a Min-Min-based job scheduling algorithm as follows. In the original Min-Min algorithm [14], the scheduler assigns jobs one at a time to the computational nodes. At each step, first it selects the unscheduled job with the smallest size and then assigns that job to the computational node having the earliest completion time after executing that job. In our proposed Min-Min-based scheduling algorithm, the required execution time considers two additional factors: the execution failure probability and the fault-tolerance mechanism.

Assigning one job to a different computational node and a different storage node has not only a different execution time but also a different failure probability. In addition, applying a different fault-tolerance mechanism for executing a job on a computational node adds up a different overhead to the total execution time. Consequently, we propose the following equations to estimate the expected execution time for executing a job, requiring to access to a storage node for data input, on a computation node protected by a fault-tolerance mechanism. We list the notations, used in the equations, in Table 1.

If JRT is applied for executing Job i on Computational Node j when Storage Node R is used for data access, we calculate the expected execution time as follows.

$$E(T_{j,R}^i) = (1 + \frac{1}{2} NP_j^i + \frac{1}{2} NP_j^{i^2} + \frac{1}{2} NP_j^{i^3}) \times \frac{SZ_i}{C_j} + DP_j^R \times WT + \frac{DS_i}{min\{B_j, RB_R\}}$$

If JMG is applied for executing Job i on Computational Node j when Storage Nodes R is used for data access, we calculate the expected execution time for the initial node as follows.

$$E(T_{j,R}^i) = (1 - \frac{1}{2} NP_j^i) \times \frac{SZ_i}{C_j} + NP_j^i \times MC_{j,k}^i + DP_j^R \times WT + \frac{DS_i}{min\{B_j, RB_R\}}$$

where

$$MC_{x,y}^{i} = \frac{D_i}{BW_{x,y}}, \forall i, x, y.$$

If JMG is adopted, the expected execution times for the two backup nodes *k* and *q* are calculated similarly except the followings. The first backup node will execute the job only when the job is failed in the initial node. The job will be executed in the second backup node only when the job is failed in the initial node and the first backup node.

Table 1. Natations used in the equations for calculating the expected execution time

$E(T_{j,R}^{i})$: the expected execution time for running job *i* on computational node *j*.
SZ_i : the workload of job *i*.
C_j : computing capacity of computational node *j*.
NP_j^i : the failure probability of running job *i* on computational node *j*.
DP_j^R : the failure probability of downloading the input data from storage node *R* for job *j*.
B_j : the bandwidth of computational node *j*.
RB_R : the remaining bandwidth of storage node *R*.
aft_j^i : average completion time if a failure occurs when running job *i* on computational node *j*.
DS_i : the input data size for job *i*.
WT : the waiting time for re-connection.
awt_j^i : average waste time if a failure occurs when running job *i* on computational node *j*.
$MC_{j,k}^{i}$: the migration cost from computational node *j* to computational node *k*.
PR : the period time of the checkpointing operation.
OH_j : the overhead of performing the checkpointing operation each time for computational node *j*.
$RM_j^i(j,k,q)$: the remaining workload of running job *i* on node *j* after checkpointing. The two backup nodes are Nodes *k* and *q*.

If JCP is applied for executing Job *i* on Computational Node *j* when Storage Node *R* is used for data access, we calculate the expected execution time for the initial node as follows. The expected execution times for the two backup nodes *k* and *q* are calculated similarly.

$$E(T_{j,R}^{i}) = (1-NP_j^i)\times\left(\frac{SZ_i}{C_j}+\left\lfloor\frac{\frac{SZ_i}{C_j}}{PR}\right\rfloor\times OH_j\right)+NP_j^i\times\left(\frac{SZ_i}{2\times C_j}+\left\lfloor\frac{\frac{SZ_i}{2\times C_j}}{PR}\right\rfloor\times OH_j+MC_{j,k}\right)$$
$$+DP_j^R\times WT+\frac{DS_i}{min\{B_j,RB_R\}}$$

where

$$RM_j^i(j,k,q) = SZ_i - \left\lfloor\frac{\frac{SZ_i}{2\times C_j}}{PR}\right\rfloor\times PR\times C_j$$
.

If JRP is applied, the expected execution time of each computational node executing a replica is calculated according to which of JRT, JMG and JCP is adopted by the node.

In our proposed scheduling algorithm, the job to be scheduled with the smallest size will be selected for next node assignment. For the job to be scheduled next, the expected execution times for executing it on each computational node supporting difference fault-tolerance mechanisms will be all calculated. The job will be assigned to the computational node that has the earliest completion time after executing the job. In addition, the selected computational node will adopt the fault-tolerance mechanism that provides the earliest completion time for executing the job on the node. To simplify the decision procedure at each scheduling step, only the expected execution time for the initial computational node is calculated according to the following equation.

$$E(T_{j,R}^{i}) = (1+\frac{1}{2}NP_j^i)\times\frac{SZ_i}{C_j}+\frac{DS_i}{min\{B_j,RB_R\}}, \quad (1)$$

Once the computational node and the storage node are determined, we consider both the job size and the failure probability to decide which fault-tolerance mechanism will be adopted for the job. We set three thresholds to help select an appropriate fault-tolerance mechanism. They are JRT, JCP and JRP thresholds and the threshold values are between 0 and 1. The JCP threshold is calculated by the following equation.

(*The largest job size in this scheduling cycle – the smallest job size*) x *JCP_rate* + *the smallest job size*

where *JCP_rate* denotes the predefined value, ranging from 0 to 1. The detailed algorithm is shown in Fig. 1. First, we test whether the failure probability is smaller than the JRT threshold, the JCP threshold and the JRP threshold in turn. Accordingly, an appropriate fault-tolerance mechanism will be selected for the job. Before scheduling the next job, we use our proposed equations mentioned above to update the expected execution times for the assigned computational node and the bandwidth allocation for the assigned storage node for the job. If backup nodes are required, their expected execution times have to be updated according to the equations.

```
Security-Aware Min-Min based Algorithm()
{
    while (Any jobs have not been scheduled yet)
    {
      select the unscheduled job that has the smallest size;
      assign the job to the computational node and the storage node that has the earliest
            completion time by equation (1);
        if (the failure probability is smaller than the JRT threshold)
          use JRT for the job;
        else
        {
          if (the normalized job size is smaller than the JCP threshold)
          {
            use JCP for the job;
            select two backup nodes based on the earlier completion time;
              if (the failure probability is smaller than the JRP threshold)
              {
                  use JCP for the replicated job;
                  select three computational nodes randomly for the replicated job;
              }
          }
          else
          {
            use JMG for the job;
            select two backup nodes based on the earlier completion time;
            if (the failure probability is smaller than the JRP threshold)
              {
                use JMG for the replicated job;
                  select three computational nodes randomly for the replicated job;
              }
          }
        }
      for the scheduled job, update the earliest completion times according to the proposed
        equations for the assigned computational nodes and the bandwidth allocation for the
        assigned storage node.
    }
}
```

Fig. 1. The proposed security-aware Min-Min based algorithm.

4 Simulation results

To evaluate our proposed job scheduling algorithm, we have constructed a simulator written in the C language. The simulation parameter settings are listed in Table 2.

We compare our proposed scheduling algorithm with the Min-Min and Sufferage algorithms [14]. The sufferage algorithm selects the unscheduled job that has the largest sufferage value at each step, where the sufferage value of a job is the difference between its second earliest completion time and its earliest completion time. The selected job will then be assigned to the computational node that has the earliest completion time for that job. For the Min-Min and Sufferage algorithms, each job will be randomly assigned one of the four fault-tolerance mechanisms.

We used the following three cases for performance comparison. In Case A, the number of instructions of a job is between 50 and 100 billion instructions. In Case B, the number of instructions of a job is between 50 and 500 billion instructions. In Case C, the number of instructions of a job is between 50 and 1000 billion instructions.

Our proposed algorithm outperforms the other two algorithms in all these three cases, as shown in Fig. 2. When the job sizes are distributed in a larger range, the makespan can be reduced more by our proposed algorithm, as shown in Fig. 2. The reason can be explained by the number of failures occurred in different algorithms as shown in Fig. 3. Our algorithm can reduce the failure rate because the high-risk jobs are allocated to more reliable computational nodes. Interestingly, although the Min-Min and the sufferage algorithms have almost the same failure rate, the sufferage algorithm is better than the Min-Min algorithm when the job sizes are distributed in a larger range.

Table 2. Simulation parameter settings

Number of jobs	2000
Number of computational nodes	50
Number of storage nodes	6
Job size	40 ~ 240 (Mb)
Job arrival time	0.0278 jobs/min/node
Job workload	50 ~ 100 (billion instructions)
Node processing speed	20 ~ 200 (million instructions/sec)
Node bandwidth	100 ~ 200 (Mb/sec)
Number of storage nodes	5
Storage node bandwidth	1000 ~ 1500 (Mb/sec)
Input data size	50 ~ 100 (Gb)
Connection failure probability	5% ~ 10%
Job security demand	0.6 ~ 0.9 (uniform dist.)
Node trust level	0.3 ~ 1.0 (uniform dist.)
Failure coefficient	λ= 3
Schedule cycle	4 hours
Checkspan	1200 sec
JRT threshold	0.4
JCP_rate	0.4
JRP threshold	0.55

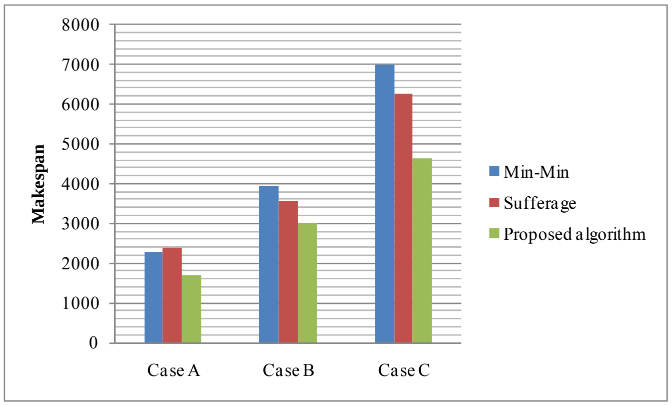


Fig. 2. Comparison of makespan.

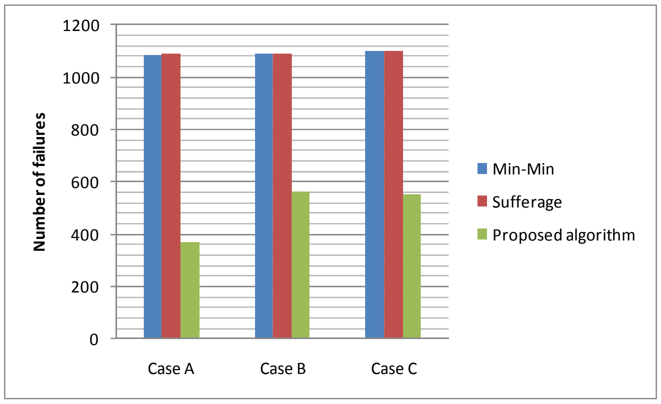


Fig. 3. Comparison of the number of failures.

5 Conclusions

In this paper we propose a job scheduling algorithm to address the problem that a large-scale grid system is unreliable by nature. This new job scheduling algorithm considers the following aspects different from our previous work. (1) The algorithm is Min-Min based, rather than genetic algorithm. Therefore the computation complexity of the new algorithm is less than that of the previously proposed genetic algorithm. Furthermore, the new algorithm has no need to rely on the scheduling history of each application to accelerate the convergence speed in the previously proposed space-time genetic algorithm. (2) The fault-tolerance mechanisms considered are the retry, the migration without checkpointing, the migration with checkpointing and the replication mechanisms. In the new algorithm, each computational node supports all four mechanisms. Our previous work assumed that each computational node supports the replication mechanisms and one of the rest three mechanisms. (3) The access to the storage node for data input is modeled in the new algorithm, which is ignored in the previous algorithm.

To provide a more reliable computation, each computational node can apply one or more fault-tolerance mechanisms to protect the execution of a job. The proposed Min-Min-based algorithm can effectively and efficiently decide which computational node(s), which data node and which fault-tolerance mechanism(s) is the best combination for executing a job. After applying the proposed algorithm, the makespan and the failure rate can be cut down significantly.

Acknowledgement

The authors would like to thank the National Science Council, Taiwan, for financially supporting this research under Contract No. NSC98-2221-E-018-008-MY2.

References

[1] I. Foster, E.C. Kesselman, The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann, San Fransisco, 2004.

[2] Globus. http://www.globus.org/.

[3] Netsolve. http://icl.cs.utk.edu/netsolve/pubs/index.html/.

[4] Condor-G. http://www.cs.wisc.edu/condor/condorg/.

[5] T.M. Project. http://www.cs.virginia.edu/~mentat/.

[6] N. Doulamis, A. Doulamis, E. Varvarigos, T. Varvarigou, "Fair Scheduling Algorithms in Grids", IEEE Transactions on Parallel and Distributed Systems, Vol.18, No.11, pp. 1630–1648, 2007.

[7] S. Viswanathan, B. Veeravalli, T. Robertazzi, "Resource-aware Distributed Scheduling Strategies for Large-scale Computational Cluster/Grid Systems", IEEE Transactions on Parallel and Distributed Systems, Vol.18, No.10, pp.1450–1461, 2007.

[8] Y. Lee, A. Zomaya, "Practical Scheduling of Bag-of-tasks Applications on Grids with Dynamic Resilience", IEEE Transactions on Computers, Vol. 56, No.6, pp. 815–825, 2007.

[9] R. Bertin, A. Legrand, C. Touati, "Toward a Fully Decentralized Algorithm for Multiple Bag-of-tasks Application Scheduling on Grids", Proc. 9th International Conference on Grid Computing, pp. 118–125, 2008.

[10] N. Nobrega, L. Assis, F. Brasileiro, "Scheduling CPU-Intensive Grid Applications Using Partial Information", Proc. 37th International Conference on Parallel Processing, pp. 262–269, 2008.

[11] S. Song, Y.-K. Kwok, K. Hwang, "Risk-resilient Heuristics and Genetic Algorithms for Security-assured Grid Job Scheduling", IEEE Transactions on Computers, Vol. 55, No. 6, pp. 703-719, 2006.

[12] S. Song, Y.-K. Kwok, K. Hwang, "Security-driven Heuristics and a Fast Genetic Algorithm for Trusted Grid Job Scheduling", Proc. 19th IEEE International Parallel and Distributed Processing Symposium, pp.65.1–65.1, 2005.

[13] C.-C. Wu and R.-Y. Sun, "An Integrated Security-aware Job Scheduling Strategy for Large-scale Computational Grids", Future Generation Computer Systems, Vol. 26, No. 2, pp. 198-206, 2010.

[14] T. Braun, D. Hensgen, R. Freund, H. Siegel, N. Beck, L. Boloni, M. Maheswaran, A. Reuther, J. Robertson, M. Theys, B. Yao, "A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems", Journal of Parallel and Distributed Computing, Vol. 61, No. 6, pp. 810-837, 2001.

[15] L. Xiao, Y. Zhu, L. Ni, Z. Xu, "Incentive-based Scheduling for Market-like Computational Grids", IEEE Transactions on Parallel and Distributed Systems, Vol. 19, No. 7, pp. 903-913, 2008.

[16] M. Li, B. Yu, M. Qi, "PGGA: A Predictable and Grouped Genetic Algorithm for Job Scheduling", Future Generation Computer Systems, Vol. 22, No. 5, pp. 588-599, 2006.

[17] R.-S. Chang, J.-S. Chang, P.-S. Lin, "An Ant Algorithm for Balanced Job Scheduling in Grids", Future Generation Computer Systems, Vol. 25, No. 1, pp. 20-27, 2009.

[18] R.-S. Chang, J.-S. Chang, S.-Y. Lin, "Job Scheduling and Data Replication on Data Grids", Future Generation Computer Systems, Vol. 23, No. 7, pp. 846-860, 2007.

[19] F. Azzedin, M. Maheswaran, "Integrating Trust into Grid Resource Management Systems", Proc. Intl Conf. Parallel Processing, pp. 47-54, 2002.

[20] M. Humphrey, M. Thompson, "Security Implications of Typical Grid Computing Usage Scenarios", Proc. High Performance Distributed Computing, pp.355-362, 2001.

[21] J. Abawajy, "Fault-tolerant Scheduling Policy for Grid Computing Systems", Proc. IEEE Intl Parallel and Distributed Processing Symp. , p. 238, 2004.

[22] M. Chtepen, F. Claeys, B. Dhoedt, F.D. Turck, P. Demeester, P. Vanrolleghem, "Adaptive Task Checkpointing and Replication: Towards Efficient Fault-Tolerant Grids", IEEE Transactions on Parallel and Distributed Systems, Vol. 20, No. 2, pp. 180-190, 2009.

[23] S. Hwang, C. Kesselman, "A Flexible Framework for Fault Tolerance in the Grid", Journal of Grid Computing, Vol. 1, No. 3, pp. 251_272, 2003.

An Artificial Immune System for Task scheduling in Grid Computing With Task balancing

Amir Massoud Bidgoli[1], **Mehdi FarokhTabar**[2], **A mir Masoud Rahmani**[3]
[1]Islamic Azad University Tehran North Branch. Tehran,IRAN
[2]Islamic Azad University Science And Research Khozestan Branch. Ahvaz, IRAN
[3]Islamic Azad University Science And Research Tehran Branch. Tehran, IRAN

Abstract : This article covers a new approach in discussion of grid scheduling. This means an economy grid using a new algorithm which from now on we call it Artificial Immune System Balancing (*AISB*). This algorithm combines the methods of clonal selection, negative selection and also uses a completely new technique for generating the first generation such as communication to computation ratio (*CCR*), ability of computational resources and prioritized tasks. Hence the algorithm can obtain acceptable results in reduced time and costs. Also, with this technique, similar results can be obtained with generation of fewer antibodies in comparison with Conventional clonal selection method.

Key words: Grid Economy, Task Scheduling, AIS Algorithm, Clonal Selection, Negative Selection.

1 Introduction

Grid computing environment is a promising platform for solving large-scale computing intensive problems. In this environment, resources are distributed geographically but in logical point of view, they are as a unit resource [1, 2]. The main goal of grid is using of various equipments with higher reliability, which is inexpensive and compatible [1]. For highest efficiency, we need a useful and right scheduler, where it's aim is to relate optimum tasks to resources. Hence heterogeneity, shared resources [3] and various users requests, will result in a complicated grid scheduling.

In 2001, buyya [4] introduced an economical framework for grid. This framework causes grid users to pay for resources used by them to the resource owners. This framework emphasizes financial cost and profits to resource owners and users. Therefore financial cost and profits plays an important role in most scheduling algorithm [5]. However, the problem is that for improving finished timing and execution costs of task, a scheduler uses different policies and goals. For economic scheduling only a few techniques have been proposed [4, 6, 7]

There are suggestions for improving scheduling in grid, using metaheuristics techniques inspired from natural rules [5,12,14,15], such as genetic, simulated annealing, ant colonies, tabu search, artificial immune systems (*AIS*) and some composite techniques. In recent years *AIS* have been used in many computing fields.

The *AIS* algorithms, inspired from natural immune systems, use a new computational and well defined recognition patterns which can be used on unusually cases. The *AIS* algorithms are categorized into different models which are: negative selection, clonal selection theory, network theory and danger theory. In recent years these techniques were used efficiently [13].

The **proposed technique** in this paper is based on directed acyclic graph (*DAG*) which uses three steps. The first step is based on communication to computation ratio (*CCR*), computation ability of a resource and also priority of nodes via PETS [8] technique which takes into account the time and cost parameters to produce the first generation. In the second step clonal selection algorithm [10] is used to improve results. Finally, the third step is to use negative selection algorithm [11] to create a balanced load between computational resources. Using the three steps will produce reasonable results for scheduling based on time and cost.

The remainder of the paper is organized as follows: In section 2, the necessary introduction theories for *DAG* scheduling are discussed. In section 3 the principles of *AIS* are discussed. In section 4 the proposed algorithm with details are discussed. In section 5, evaluation, performance and comparison of proposed algorithm with several other conventional algorithms are discussed by simulation. Finally produced results are discussed with details in section 6.

2 Scheduling Problem

The directed acyclic graph (*DAG*) is introduced with $G = (V,E)$, Where V is a set of graph nodes and E is set of graph edges showing the relationship between the nodes in a graph. For example $(i,j) \in E$ shows the relationship between the node n_i and n_j. To start the task of n_j, at first n_i should be completed. Assigned to each edge a weight to show the required time for transferring the output results from node n_i to n_j.

Every *DAG* has entry node, n_{entry}, and an exit node, n_{exit}. In resource scheduling problem, a start node as entry node and an end node as exit node exists. However if more than one entry node and exist, a hypothetical node will be specified as entry node or exit node with zero weight.

In this grid, $M = \{M_{j;}\ j= 1...m\}$ is defined as available computational resources set for tasks execution. The matrix ,$W_{i\times j}$, shows the execution time of graph nodes on computational resources in

M set. An element , W_{ij} ,shows the execution time of n_i on M_j computational resource. $\overline{W}_i$ Shows the average execution time of node on computational resource and is defined by:

$$\overline{W_i} = \sum_{j=1}^{m} \frac{W_{i,j}}{m} \tag{1}$$

Matrix $MC_{m\times m}$ shows transfer rate of data between two computational resources. The vector L (m-dimensional) shows the required time for start of communication with computational resources. *The function* WC_{ij} is transfer timing of data between two nodes n_i (on machine M_p) and n_j (on machine M_q) is defined by:

$$WC_{i,j} = L_p + \frac{Data_{i,j}}{MC_{p,q}} \tag{2}$$

Where $Data_{i,j}$ is rate of data transfers from *node* n_i to n_j. If the two nodes are on one computational resource, the transfer rate is ignored and therefore $WC_{i,j}$ would be equal to zero. The function $\overline{WC}_{i,j}$ is average time of data transfer between two nodes *(i, j)* and is defined by:

$$\overline{WC_{i,j}} = \overline{L} + \frac{Data_{i,j}}{\overline{MC}} \tag{3}$$

Where $\overline{MC}$ is the average transfer rate of data between computational resources and $\overline{L}$ is the average starting time of computational resource.

The matrix, $C_{i\times j}$, defines the execution cost of graph nodes on existing computation resources in M set. The $C_{i,j}$ specifies cost of executing node n_i on the computational resource M_j. The $\overline{C}_i$ specifies the average cost of executing node n_i on computational resource which is defined by:

$$\overline{C_i} = \sum_{j=1}^{m} \frac{C_{i,j}}{m} \tag{4}$$

The $MCC_{m\times m}$ is a Matrix, showing data transmission costs per second between two computational resources and the LC vector showing the rate of cost for communication initiation with computational resources. $CC_{i,j}$ is cost of trasferring data between two nodes n_i (placed on M_p) and n_j (placed on M_q) that is defined by :

$$CC_{i,j} = (L_p * LC_p) + (\frac{Data_{i,j}}{MC_{p,q}} * MCC_{p,q}) \tag{5}$$

Average cost of relation between two nodes *(i, j)* is calculated by:

$$\overline{CC_{i,j}} = (\overline{L} * \overline{LC}) + (\frac{Data_{i,j}}{\overline{MC}} * \overline{MCC}) \tag{6}$$

$\overline{MCC}$ Is average cost of data transmission per second and $\overline{LC}$ is average cost of communication initiation with computational resources.

The $EST(n_i,M_j)$ and $EFT(n_i,M_j)$ are the earliest execution start time and earliest execution finish time of task n_i on computational resources M_j respectively. The two functions $EST(n_i,M_j)$ and $EFT(n_i,M_j)$ are shown in relations (7) and (8).

$$EST(n_i, M_j) = \max\{avail[j], \max_{n_p \in pred(n_i)} (AFT(n_p) + WC_{p,i})\} \tag{7}$$

$$EFT(n_i, M_j) = W_{i,j} + EST(n_i, M_j) \tag{8}$$

If n_i is entry node:

$$EST(n_{entry}, M_j) = 0 \tag{9}$$

Where *avail[j]* is the earliest time at which computational resource M_j is ready for task execution and $pred(n_i)$ is the set of immediate predecessor tasks of task n_i. The inner max block in the *EST* equation returns the *ready time*. (i.e., the time when all data needed by n_j has arrived at computational resource M_j).

The actual earliest start time and finish time of task n_i on computational resource M_j is equal to $AST(n_i)$ and $AFT(n_i)$. Finish time of scheduling is $AFT(n_{exit})$ that shows end of scheduling and displayed by *SL* and also sum of cost scheduling showing with *SC*. Equations (10) and (11) shows these:

$$ScheduleLenght(SL) = \max\{AFT(n_{exit})\} \tag{10}$$

$$ScheduleCost(SC) = \sum_{i=1}^{countoftasks} (\underset{\substack{j=selected\\resource}}{C_{i,j}} + \sum_{p}^{} \underset{n_p \in pred(n_i)}{CC_{p,i}}) \tag{11}$$

The communication to computation ratio (*CCR*) is a measure that indicates whether a task graph is communication intensive, computation intensive or moderate. If *CCR* is calculated based on time it is called *CCRW* and if it is calculated based on cost it is called *CCRC*. The calculation of *CCRW* is carried out as average communication time divided by average computation time and calculation of CCRC is carried out as average communication costs divided by average computation cost.

3 Particles of Immune Systems

AIS are distributed adaptive systems for problem solving using models and principles derived from the Human Immune System. The Immune System is the defense system of our body, which can produce and secrete antibodies used to protect us against infection through an antigen recognition process. Many different AIS algorithm models have been built, including bone marrow models, thymus models, clonal selection algorithms, negative Selection, and immune network models [13].

3.1 Negative Selection Algorithm

The main purpose of negative selection technique is determination of a limited affinity threshold for detection of self cells. The way it is done is that when cells called T cells are produced and mutated by thymus, those T cells that are capable of detecting self antigens are separated from the colony and the rest of T cells for detecting non-self antigens are entered into blood stream. The above process is called negative selection. Figure (1) shows function of negative selection algorithm [11]:

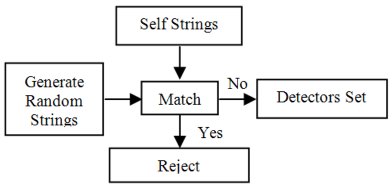


Fig. 1. Detector Set Generation of a Negative Selection Algorithm [11]

3.2 Clonal Selection Algorithm

When a cell called B cell is stimulated, it will quickly duplicate itself. While it is duplicating , a gene mutation proportional to the inverse degree of affinity is produced which causes the production of cells with higher degree of affinity. This process is called clonal selection. Figure (2) shows function of clonal selection algorithm. In this algorithm,

(1) Generate a set (*P*) of candidate solutions, composed of the subset of memory cells (*M*) added to the remaining (*Pr*) population (*P* = *Pr* + *M*);

(2) Determine (Select) the n best individuals of the population (*Pn*), based on an affinity measure;

(3) Reproduce (Clone) these n best individuals of the population, giving rise to a temporary population of clones (*C*). The clone size is an increasing function of the affinity with the antigen;

(4) Submit the population of clones to a hypermutation scheme, where the hypermutation is proportional to the affinity of the antibody with the antigen. A maturated antibody population is generated (*C**);

(5) Re-select the improved individuals from *C** to compose the memory set M. Some members of P can be replaced by other improved members of *C**;

(6) Replace d antibodies by novel ones (diversity introduction). The lower affinity cells have higher probabilities of being replaced [10].

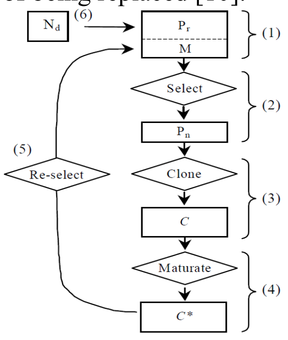


Fig. 2. Diagram of Clonal Selection Algorithm [10]

4 Artificial Immune Systems with Balancing (AISB)

The Suggested algorithm in this article, ***AISB***, includes three steps as follows: (1) prioritizing tasks and computational resources for the primary generation production, (2) clonal selection and (3) balancing by selection negative algorithm.

4.1 Priority Step

In this step, The PETS algorithm is combined with cost and time parameters for tasks priority and the two parameters can be adjusted by user. Also, prioritizing of computational resources based on cost and time is carried out.

4.1.1 Task Priority Step

Each graph divided to several levels, entry node in first level and exit node in last level. generally, the node n_i is placed in K-level provided for all edges (n_i, n_j), the node n_j was placed in a lower level than k and there should be a minimum of one edge between n_i and n_j and lower level must be one level below the level k(i.e. level k-1).

In PETS algorithm, for assigning priority to a node three parameters called Average Computation Cost *(ACC*), Data Transfer Cost (*DTC*) and Rank of Predecessor Task (*RPT*) are defined. In this paper by manipulating the above mentioned parameters, new formulae with better efficiencies are obtained. From now on, the above parameters are called Average Computation Cost and Time (*ACCT*), Data Transfer Cost and Time (*DTCT*). The parameter *ACCT* is defined as evaluation of cost and time of a node on computational resource *M* which is calculated as:

$$ACCT(n_i) = T_p * \frac{\overline{W_i}}{W_{\max}} + C_p * \frac{\overline{C_i}}{C_{\max}} \qquad (12)$$

Where T_p and C_p are coefficient of time and cost importance in a user view which can have a value between 0 and 1. W_{max} and C_{max} are maximum time and cost of a finished task on a computational resource. The parameter *DTCT* is defined as evaluation of time and cost of communication related to data transfer of a node to all the immediate nodes. This parameter is calculated in level 1 as:

$$DTCT(n_i) = T_p * \frac{\overline{WC_{i,j}}}{WC_{\max}} + C_p * \frac{CC_{i,j}}{CC_{\max}} \qquad (13)$$

Where, WC_{max} and CC_{max} are the maximum time and cost of node communication. The parameter *RPT* is defined as the highest degree of predecessor nodes and calculated by:

$$RPT(n_i) = \max\{rank(n_1) + ... + rank(n_h)\} \quad (14)$$

Where n_1, n_2,…, n_h are immediate predecessor nodes. For the entry node the value of RPT is zero (i.e. $RPT(n_{entry})$=0). The degree of priority for a node n_i is based on *RPT*, *DTCT* and *ACCT* values and calculated by:

$$rank(n_i) = round(ACCT(n_i) + DTCT(n_i) + RPT(n_i)) \quad (15)$$

Each level has its own degree of priority for its nodes and a node with the highest degree of priority, has the highest priority on its level. This procedure is repeated for all levels in a graph. For example the gragh of figure (3) shows the priority of tasks in table 1.

4.1.2 Computational Resource Priority Step

Priority of computational resources will cause the most suitable computational resource (time and cost efficient) be assigned to a special node. For priority determination of a computational resource the parameters, time and cost of execution, T_p, C_p, CCRW and CCRC are used.

The new Matrix $EM_{i\times j}$ is defined as the evaluation of tasks on computational resources. So that $EM_{i,j}$ shows the evaluation of executing n_i on computational resource M_j and calculated by:

$$EM_{i,j} = \frac{1}{T_p * \frac{W_{i,j}}{W_{\max}} + C_p * \frac{WC_{i,j}}{WC_{\max}}} \qquad (16)$$

The new ECCR coefficient is defined as the evaluation of communication and computation intensity with respect to T_p and C_p parameters and calculated as:

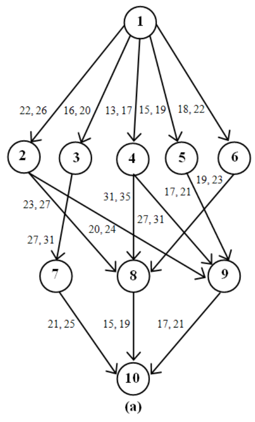
$$ECCR = T_p * CCRW + T_c * CCRC \qquad (17)$$

Finally, the matrix of computational resource priority coefficient, $EMC_{i\times j}$, with condition of ECCR>=1 and ECCR<1 are respectively calculated as:

$$EMC_{i,j} = EM_{i,j} + (ECCR * (EM_{i,j} - \min(EM_{i,1..m})) \qquad (18)$$

$$EMC_{i,j} = EM_{i,j} - ((1 - ECCR) * (EM_{i,j} - \min(EM_{i,1..m})) \qquad (19)$$

The output matrix $EMC_{i\times j}$ shows the priority of computational resource per task so that the highest priority computational resource is most probably will be chosen.



Task	M1	M2	M3	Level	Priority
1	14	16	9	1	1
2	13	19	18	2	1
3	11	13	19	2	3
4	13	8	17	2	2
5	12	13	10	2	5
6	13	16	9	2	4
7	7	15	11	3	3
8	5	11	14	3	2
9	18	12	20	3	1
10	21	7	16	4	1

(b)

Resource	Cost
M1	0.75$
M2	0.85$
M3	1$

(c)

Fig. 3. (a) A sample graph with 10 tasks (First Number is Average of Time Communication $\overline{WC}$, Second Number is Average of Cost Communication $\overline{CC}$). (b) Time matrix and priority of tasks on resources. (c) table of computation cost. User coefficient (T_p = 0.5, C_p=0.5)

4.2 Clonal Selection Step

To improve time and cost of tasks in grid, the suggestive clonal selection algorithm is used and implemented. This proposed algorithm is shown in figure (4).

```
1: Compute priority for tasks
2: Compute priority for resource
3: Generate initial antibody population, AB Base on Priority
4: for each generation ($N_g$) do
5:   Select a subset of AB with the highest calculated affinity as $AB^*$
6:   for each antibody $ab_i$ in the antibody population $AB^*$ do
7:     Clone $ab_i$ proportional to its affinity and $N_g$
8:     for each clone $c_{ij}$ in the clone set $C_i$ do
9:       Generate a set M inversely proportional to abi's affinity and $N_g$
10:      for each random schedule $m_k$ in M do
11:        Mutate $c_{ij}$ with $m_k$
12:        Sort the tasks of $c_{ij}$ according to the tasks priority
13:        Compute the affinity of $c_{ij}$:
14:      end
15:    end
16:    Set $ab_i$ to its best clone whose affinity is higher than abi's
17:  end
18: Replace Highest $AB^*$ with antibody population, AB
19: Replace worst b% of antibodies in AB by randomly generated ones
20: end
```

Fig.4. Proposed technique based on clonal selection Algorithm

In figure (4), lines 1 and 2 show the priority of tasks and computational resources based on previously defined relations. In line 3, the primary generation based on priorities is produced. For selection of computational resources, the method of roulette wheel is used such that the resources with higher priorities are most probable to be selected. In the line 5, a subset of the best antibodies with respect to limited threshold affinity based on the newly defined function given blow is selected.

$$select = \max\{0.2 * |AB|, (\frac{N_G}{NG_S}) * |AB|\} \qquad (20)$$

In the formula (20) the set of antibodies are assumed to be sorted according to the threshold affinity. In the formula, NG shows the current generation of computations and NG_s shows the number of generations produced by clonal selection algorithm. The parameter, |AB|, determines the total number of antibodies. In the select formula (20), as we get near to final generations, the number of selected antibodies will be reduced which causes the algorithm to be more concentrated on the best antibodies and also will cause the algorithm running time to decrease considerably. In line 7, the number of clone (*NC*) calculates the antibodies based on the number of current generation and rate of its threshold affinity which is shown in equation (21).

$$NC = \max\{2, ((\frac{N_G}{(0.25 * NG_S) * |AB|}) + |AB|) * (1 - Affinity(SSL, SL(ab_i) - 1))\} \quad (21)$$

Where *SSL* parameter determines the best threshold affinity (i.e. best in terms of time and cost of execution) in equation (21), as the rate of threshold affinity and number of generation is increased the number of antibody clone will be increased. In the line 9, the total number of idle slots in computational resources are determined and then the number of mutations, NM, based on the idle slots are calculated which is shown in equation (22).

$$NM = \max\{4, ((\frac{NG_S - N_G}{(0.33 * NG_S) * |AB|}) + 2|AB|) * (Affinity(SSL, SL(ab_i) - 1))\} \quad (22)$$

Where SSL parameter determines the best threshold affinity (i.e. best in terms of time and cost ofexecution.

In equation (22), as rate of threshold affinity and number of generations increases, the number of mutations decreases. The rate of threshold affinity is calculated by the time and cost of current antibody relative to the best antibody as shown in equation (23).

$$Affinity(ab_i) = (C_p * (\frac{SC_i}{SC_{best}})) + (T_p * (\frac{SL_i}{SL_{best}})) \quad (23)$$

In lines 11, 12 and 13, mutations were made on antibodies and the antibodies are sorted based on priority of tasks and then the threshold affinity are calculated. In lines 16, 18 and 19, the best mutated antibodies replaced the previous ones and then about b% (default 20%) from the worst antibodies of each generation are replaced with new ones which are produced randomly.

4.3 Use Of Negative Selection For Balancing

To balance tasks execution on computational resources using negative selection, the worst computational resources movement are chosen and then by replacement (e.g. the computational resources which has the highest tasks running on it) and placing current tasks on idle slots of other computational resources, a balance between computational resources are produced. This algorithm is shown in figure (5).

```
 1: Select Best s% of antibodies in AB by highest affinity as $AB^*$
 2: **for** each antibody $ab_i$ in the antibody population $AB^*$ **do**
 3:   **for** each Resource $r_i$ in $ab_i$ **do**
 4:     **if** affinity $r_i$ is higher than affinity threshold **then**
 5:       **for** each Task $t_i$ in $r_i$ **do**
 6:         Replacement $t_i$ with task that same level
 7:         Insert $t_i$ in the best place of other resources
 8:         Insert $t_i$ in the idling slots of resources
 9:       **end**
10:     **end**
11:   **end**
12: **end**
13: Replace Highest $AB^*$ with antibody population, AB
```

Fig.5. Proposed technique based on negative selection algorithm

In figure (5), lines 1, S% (default 25%) of total number of antibodies are selected based on threshold affinity. In lines 3, 4 and 5, for every antibody most busy computational resource whose affinity is more than an average threshold affinity, is selected. In this algorithm, affinity threshold means the average time or cost of computational resource usage. And in lines 6,7 and 8, the ordered operations of selected task is exchanged with same level tasks, placing the selected task in the best place amongst computational resources and inserting a selected task in the idle slots of computational resources. The lines 6, 7 and 8 for each task in selected computational resource is repeated. The last line shows the best antibodies are chosen based on threshold affinity and is replaced by the main generation. This operation is repeated for every generation of clonal selection algorithm.

5 Simulation Performance results

In this section, the classic *AIS* clonal selection algorithm is compared with our proposed modified *AIS* clonal selection algorithm and then, the simulation result of proposed algorithm ,*AISB* ,for task scheduling on grid computation are presented .With respect to the dependency of functions on cost and time, the obtained results based on cost and time are compared with two other methods. The first scheduler is HEFT [9] and the second scheduler is classic clonal selection algorithm.

Here, result related to compare of efficiency of algorithms based on time and cost are obtained by using normalized schedule length (*NSL*) and normalized schedule cost (*NSC*). The normalized schedule length and schedule cost is defined to be schedule length and schedule cost obtained by a particular algorithm over schedule length and schedule cost obtained by the HEFT algorithm.

The values of parameters used for our experiments are: number of generations {3, 5 and 10}, number of antibodies {5, 10 and 20}, elimination rate 20%, number of computational machine was {3,6,16,24} and used with 100,200 and random speed, number of task

{10,30,45,80,180} and *CCR* coefficient was with rate of {0.1,1,10}. Also as cost and time parameters are optional for a user, for comparison purposes we need to compare with cost (i.e. $T_p=0$ and $C_p=1$) once and compare with time (i.e. $T_p=1$ and $C_p=0$) once.

In figure (6), a simulation results between classic *AIS* running algorithm and our modified *AIS* running algorithm is shown. As can be seen from figure (6), the modified clonal selection algorithm with control over the number of antibodies, clonal and mutation based on the number of generations and also task and computational resource prioritization, could produce better results because of reduced number of antibodies production (Hence, the running time of the algorithm reduces). The result shown in figure (7) indicates the better performance of proposed algorithm in comparison with AIS and HEFT algorithms for the execution timing of tasks. This is because in periods when communication intensity is increased, the type of tasks and computational resources prioritization and insertion of tasks in idle slots are carried out. Also when computational intensity is increased, the used balancing algorithm produce the above mentioned better performance.

In figure (8), the result of execution costs is shown which are better than other algorithms. The reason for better results are the use of balanced algorithm and control of threshold affinity relation.

6 Conclusion And Recommendation

In this article, a new algorithm ,*AISB* ,for task scheduling based on DAG is presented. The proposed algorithm, with respect to user requirements, determines the importance of cost and time. The purpose of proposed scheduler is to reduce execution time and cost based on mentioned parameters. In this scheduler, priority of computational resources and tasks are determined by time and cost parameters and then for scheduling optimization, the clonal selection algorithm is used. Next, by using the negative selection algorithm, balancing between computational resources based on cost and time parameters are carried out. Finally the simulation results of three schedulers are compared. The result obtained show that the proposed algorithm in situations where CCR coefficient is low (computational intensity) or is very high (communication intensity) have a better performance.

It can be recommended to use the scheduler in situations where the rate of intensity of processing and communication are similar to see if optimization can be achieved.

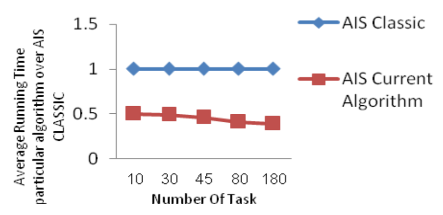


Fig.6. Average Running Time a particular algorithm over AIS Classic

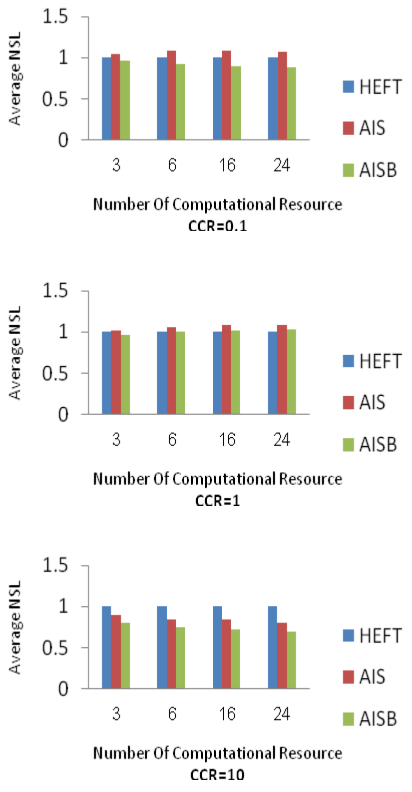
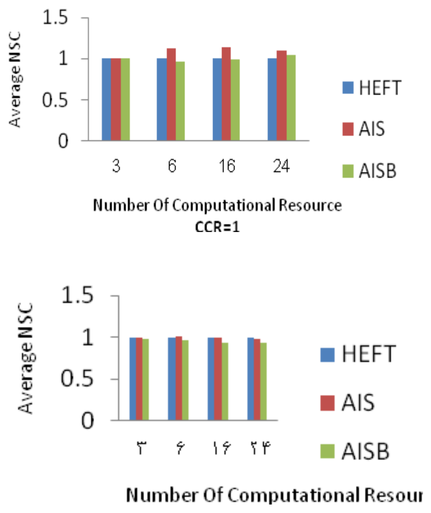


Fig.7. Average NSL of random DAGs



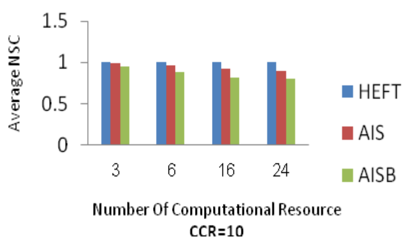


Fig.8. Average NSC of random DAGs

7 Refrences

[1] I. Foster and C.Kesselman (editors), The Grid 2: blueprint for a new Computing Infrastructure, Morgan Kaufmann Publishers, USA, 2004.

[2] Baker M., Buyya R., and Laforenza D., *"The Grid: International Efforts in Global Computing",* Proc. Of International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet, Rome, Italy, 2000.

[3] M, Arora, S.K.Das, R. Biswas, *A Decentralized Scheduling and Load Balancing Algorithm for Heterogeneous Grid Environment,* in Proc of International Conference on Parallel Processing Workshops (ICPPW'02),pp.:499-505, Vancouver, British Columbia Canda, August 2002.

[4] R. Buyya, J. Giddy, D. Abramson, "A case for economy grid architecture for service-oriented grid computing", in: 10th IEEE internet Heterogeneous Computing Workshop (HCW 2001), San Francisco, CA,April 2001.

[5] L. Young, S. McGough, S. Newhouse, and J. Darlington, *Scheduling Architecture and Algorithms within the ICENI Grid Middleware,* in Proc. Of U"K e- science All Hands Meeting, pp. 5-12, Nottingham, UK, September 2003.

[6] R. Sakellariou, H. Zhao, E. Tsiakkouri, M. D. Dikaiakos. "*Scheduling Workflows with Budget Constraints*". In S.Gorlatch, M.Danelutto (Eds.), Integrated Research in Grid Computing, CoreGrid series,
Springer-verlag, to appear 2005.

[7] J. Yu,R.Buyya C. Khong Tham. "Cost-based scheduling of scientific workflow applications on utility grids",vic.,Australia;,e-Science and Grid Computing, 2005 First International Conferece on,Dec 2005.

[8] E. Ilavarasan and P.Thambidurai. "Low Complexity Performance Effective Task Scheduling Algorithm for Heterogeneous Computing Enviroments". Journal of Computer Sciences 3 (2):94-103,2007

[9] H. Topcuoglu, S.Hariri, and M. Wu. "Performance-effective and low-complexity task scheduling for heterogeneous computing". In IEEE Transactions on Parallel and Distributed Systems, volume 13(3) 2002.

[10] L.N. de Castro and F.j. Von Zuben. "The Clonal Selection Algorithm with Engineering Applications". In Proceedings of the Genetic and Evolutionary Computational Conference, pages 36 – 37, 2000.

[11] J.Kim and P.J. Bentley. "An Evaluation of Negative Selection in an Artificial Immune System for Network Intrusions Detection". In Proceedings of the Genetic and Evolutionary Computation Conference, pages 1330-1337, 2001.

[12] Young Choon Lee and Albert Y.Zomaya, "An Artificial System for Heterogeneous Multiprocessor Scheduling with Task Duplication", IEEE International Parallel and Distributed Processing Symposium, 2007.IPDPS 2007.

[13] L. N. de Castro, J. L Timmis "Artificial immune systems as a novel soft computing paradigm" soft computing 7 (2003) 526-544 springer-verlag2003.

[14] R. Braun, H. Siegel, N. Beck, L. Boloni, M. Maheswaran, A. Reuther, G. Robertson, M. Theys, B. Yao, D. Hensgen and R. Freund, A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems, in J. of parallel and Distributed Computing, vol.61, No. 6, pp. 810-837, 2001.

[15] Anna Swiecicka, Franciszek Seredynski, and Albert Y. Zomaya, "Multiprocessor Scheduling and Rescheduling with Use of Cellular Automata and Artificial Immune System Support", IEEE Transactions on Parallel and Distributed Systems, Vol 17, No. 3,March 2006.

Towards a Neuro-Dynamic Technique for Optimal Scheduling and Allocation of Resources in a GUISET Enterprise Grid.

Ekabua, Obeten O., Dzawo, Gilbert and Soganile, Ndabezinhle.
Department of Computer Science and Information Systems
University of Venda, Private Bag X5050, Thohoyandou 0950, South Africa.

{obeten.ekabua@univen.ac.za, dzawog@univen.ac.za, ndabezinhle.soganile@univen.ac.za}

Abstract: *Allocation and scheduling of resources within a Grid infrastructure is an NP-Complete optimization problem. Combinatorial optimization problems are either NP-hard or NP-Complete and the theory of NP-Completeness has reduced hopes that NP-hard problems can be solved within polynomial bounded computation times. Deciding which jobs are allocated to which resources has continued to generate serious concerns and challenges to researchers in the field, and the task of scheduling computing resources remains very difficult. This is because resources are geographically distributed and owned by individuals with different access and cost policies. More so, the existing static allocation mechanisms are inept to handle the dynamically changing characteristics of the grid resources. Therefore, the need for new resource allocation methods or modification of the existing ones is inevitable. In this paper, we propose a service management model with a neuro-dynamic programming technique, for optimal resource allocation and scheduling in a Grid-based utility infrastructure for SMME's Enabling Technology (GUISET).*

Keyword Words*: Scheduling, Optimal Resource Allocation, Service Management, Grid Computing. GUISET, Neuro-Dynamic programming.*

1. Introduction.

With an increasing technology evolution that is combined with web revolution, new challenges in the context of service provisioning has also emerged. Moreso, with an increasing use of the internet as a distributed environment for service provisioning, human interaction with computers has changed as they relate to their point of presence within the service provisioning environment [1].

Computational Grids have become an interesting environment for the execution of large-scale resource demanding applications. One key motivating issue for constructing Grids is to create a neuro application-level environment, between various distributed systems, so that resources and services supported by these individual systems can be shared globally [2].

Grid [3,4,5] are resource sharing and coordination infrastructures for building dynamically constructed problem-solving environments using geographically and organizationally dispersed, high-performance computing and data handling resources. Grid computing eliminates the resource islands in the application level and makes computing and services ubiquitous. The widespread adoption of the Grid computing paradigm has made the Grid an enabler for many scientific, engineering, industrial as well as commercial enterprises to create a common IT infrastructure that can be shared by business processes, delivering a higher quality service at much lower cost [6].

The task of scheduling resources is very difficult as resources are geographically distributed and owned by individuals or organizations with different access and cost policies. Scheduling and Resource management remains one of the major areas of research among others that the Global Grid Forum (GGF) [7] is actively pursuing a standard.

The research groups of the GGF working on Scheduling Optimization [8] led by V. Di Martino and E. Talbi, and Grid Resource Allocation Agreement Protocol [9] led by J. Maclaren, V. Sander and W Ziegler have respectively proposed defining measures of scheduling algorithm performance to foster the development of Grid-wide scheduling methodology. This is in addition to available schedulers and defining interactions between a higher level service and a local resource management system. The main goal of these is to facilitate the allocation and reservation of resources.

Several researches that has been done in relation to resource allocation, considered centralized resource and task admission control [10, 11, 12]. Dynamic task assignment to heterogeneous computing resources is studied in [13, 14]. Neural computing (a paradigm of artificial intelligence) is a concept that lends itself well to the heuristics of learning from experience. This is because of its ability to imitate the skill of human experts by capturing knowledge, generalizing non-linear functional relationship, and provides a flexible way of handling complex and intelligent information processing problems. Artificial Neural Networks (ANNs) have been shown to be effective as computational processors for various tasks including combinatorial optimization problems. Developing a neuro-dynamic approach for optimal scheduling and allocation of resources is an NP-Complete problem.

2. Related Work

The Master-Worker paradigm [16] addresses the problem of scheduling master-worker applications on computational grid, and presented a framework that allows the development of a tailored scheduling strategy. The idea reported in this work is said to be a simple but effective scheduling strategy that measures execution times of task dynamically and uses the information to also dynamically adjust the number of workers to achieve a desired efficiency.

Matchmaking [17], a distributed resource management mechanism, which includes several components of a resource discovery mechanism, was developed based on the idea that resources providing services and clients that request services should use a classified advertisement to advertise their characteristics and requirements. The work introduced a new class of data dissemination for resource discovery in grid systems.

NetSolve [18], a client-agent-server enhances a remote approach for users to compute complex scientific problems, as agents does the scheduling by searching for those resources that offers the best performance in a network.

AppLes [19], focuses on scheduling at the application level through the development of scheduling agents for parallel metacomputing applications where, in order to determine schedules, agents must consider the requirements of the application, the predicted load and availability of the system resources at scheduling time.

GUISET [20] is a Grid-based utility infrastructure that emanated from the product-line reference architecture (see fig. 1) of the M-commerce software infrastructure project currently under development at the Department of Computer Science, University of Zululand as an enabling technology for SMME’s. The SMME’s are here seen as ensemble of virtual organizations (VOs), reflecting dynamic collection of individuals, cooperative institutions and computational resources. One of the major goals GUISET seeks to achieve is the reduction or minimization of operating overhead cost. This goal is envisaged achievable in this proposed model by integrating utility and service management principles into our proposed optimization module.

In the context of Grid resource control, a ‘service’ is defined as any entity that exposes a useful function to its client [16, 20]. The effective management of resources and services is therefore imperative and holds substantial benefits. In view of the envisaged increasing requirements for resource sharing of computational resources, distributed over a wide area of networks within the GUISET framework, we propose an optimal resource allocation and scheduling module implementing a neuro-dynamic programming paradigm (see fig 2) within GUISET for efficient service management.

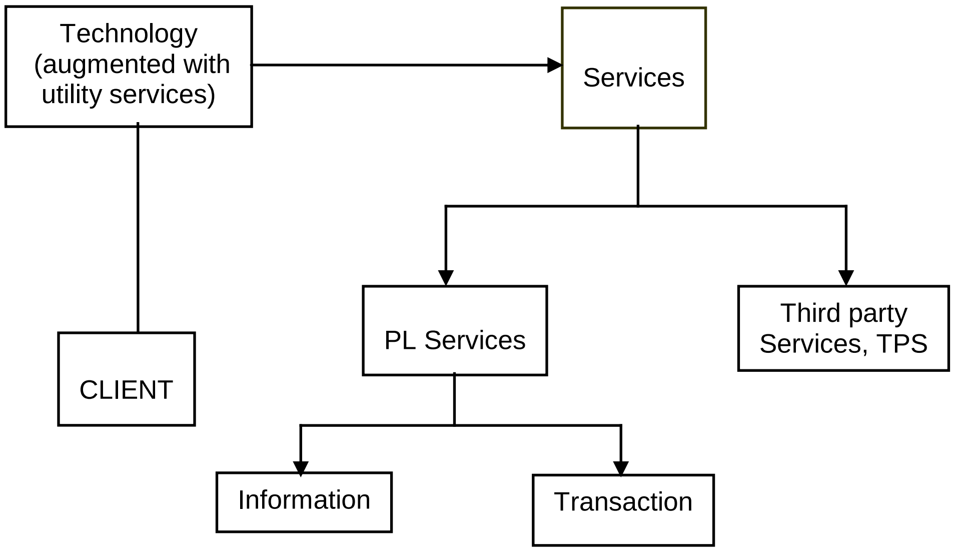

Technology (augmented with utility services)
Services
PL Services
Third party Services, TPS
CLIENT
Information
Transaction

Fig. 1: Reference architecture for the Mobile Commerce Product Line [20]

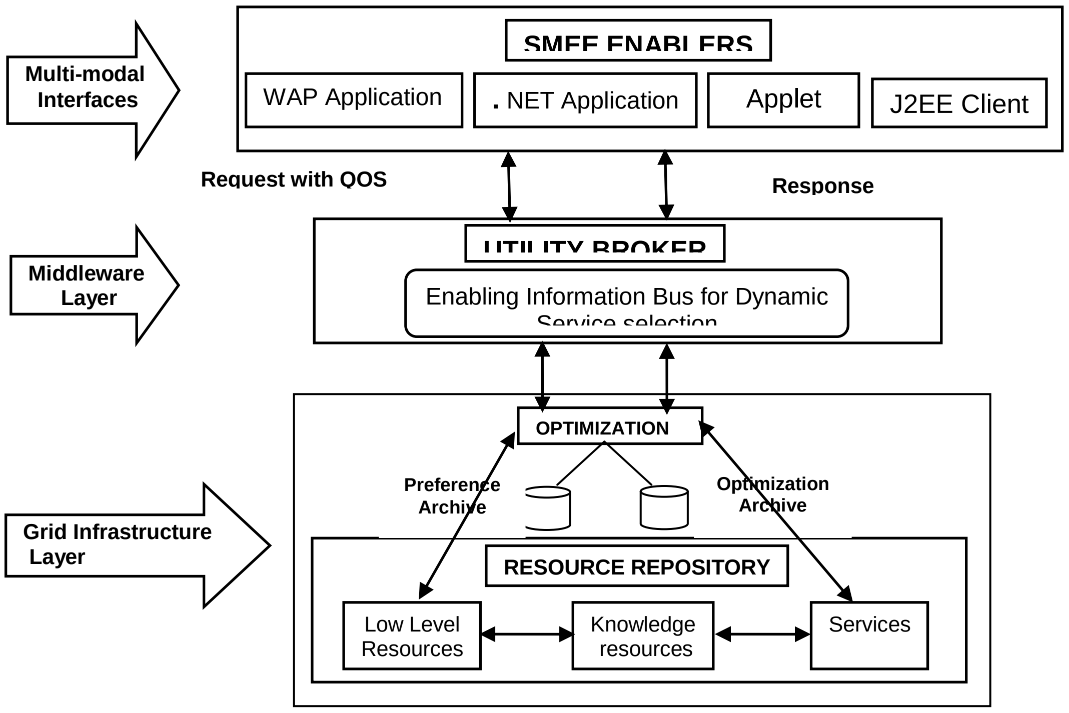


Fig. 2 GUISET Architecture with an optimization module [20].

It is believed by many researchers in the field that neural network models offer the most promising unified approach to building truly intelligent computer systems; and that the use of distributed, parallel computations as performed in ANNs is the best way to overcome the combinatorial explosion associated with symbolic serial computations when using Von Neuman Computer Architecture[21]. ANNs have been used to solve a number of problems that require finding optimal solutions.

Marbach et al 22] used a Neuro-Dynamic Network to solve a “Call Admission Control and Routing Network problem”. In [23] Tank and Hopfield used a Dynamic Recurrent Neural Network to solve the traveling salesman problem.

3. Neuro-Dynamic programming problem formulation

Definition: Given a set of P jobs where job p_i has length l_i and Q available number of resources, how do we allocate or what is the minimum possible time required to schedule all the jobs in P on Q resources?

Formulation: In order to formulate the objective function and task allocation constraints, it is necessary to define the following useful notations.

P = Number of jobs

Q = Number of available resources

$\mathbf{t_o}$ = System start time

$\mathbf{t_b}$ = Current time before allocation

$\mathbf{t_a}$ = Time after allocation

$\mathbf{j_a}$ = Job arrival rate

$\mathbf{a_p(t)}$ = Available processing capacity of resources at time t

$\mathbf{a_b(t)}$ = Available processing capacity of a particular resource at time t

$\mathbf{j_s}$ = The set of jobs to be allocated to resources

$\mathbf{D_s}$ = Duration needed to perform j_s

$\mathbf{S_i}$ = The set of task to be allocated to a particular resource

If the set of jobs j_s can be successfully allocated to the **Q** available resources, then the following relationship holds:

1. $J_s = \bigcup_{i=1}^{N} s_i$, Ensuring that all tasks are allocated

2. $\bigcup_{i \neq j}^{N} s_i s_j = \alpha \quad \forall\, i, j \leq N$ Every task is uniquely assigned to a resource.

This resource allocation problem can be represented as a linear programming (LP) model. If we let C_{ij} to be the cost associated to assigning task i to resource j, we can define the LP model as:

$$Min z = \sum_{i=1}^{n}\sum_{j=1}^{m} C_{ij} P_{ij} \qquad (1)$$

Subject to the following constraints:

i. schedulability constraint

$$a_p(t) \geq \sum_{j=1}^{m} P_{ij} = 1, i = 1, 2, \ldots, n \qquad (2)$$

ii. bounds on available resource

$$a_b(t) \geq \sum_{i=1}^{n} P_{ij} = 1, j = 1, 2, \ldots, n \qquad (3)$$

the LP model can be reduced to the matrix form $Ap = b$ which can be solved iteratively using neuro-dynamic programming for optimal solution. Fig. 3 is a typical feedforward network capable of handling such problem.

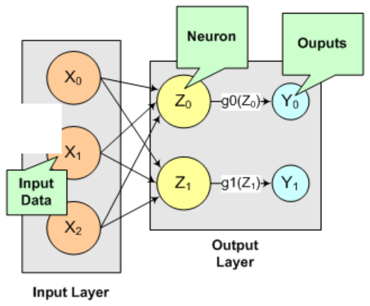


Figure 3: An example of a single layer feedback Neural Networks [22].

4. Our Proposed Approach

In order to adequately address issues of allocation and scheduling, we propose to employ the following steps in tackling the problem:

Step 1: **Model formulation of problem domain**. Modeling is a critical step to problem solving and good models will yield good and efficient problem solving strategy [16].

Step 2: **Use of fact-finding technique**. This is required because resource sharing within a grid infrastructure or virtual organizations (VO) requires the availability of rich information support system and service discovery mechanism to facilitate a very good decision making process. The findings will embrace resource availability, performance, usage and management of heterogeneous resources.

Step 3: **Activation of GGF techniques involving 3 phases.** In this step, the ten steps for Job scheduling proposed by GGF [7] (fig. 3) shall be critically analyzed, and built into our conceptual mathematically optimized model.

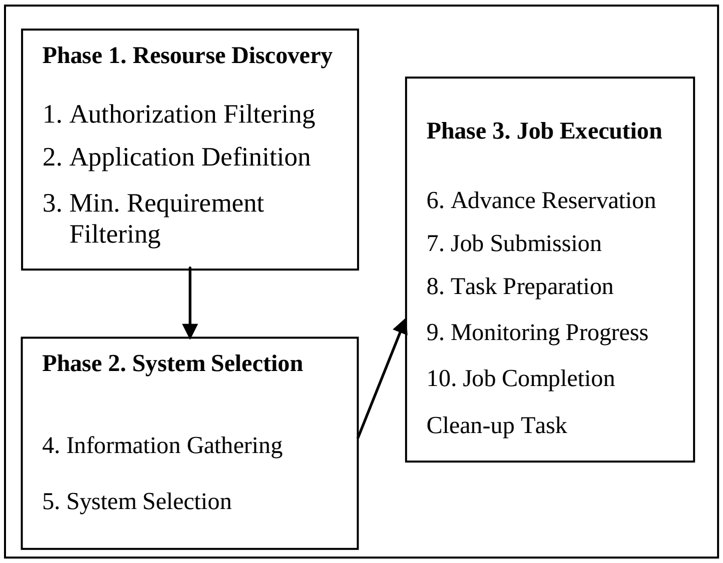


Fig. 4: Ten steps for job scheduling [7].

Step 4: **Development of neuro-dynamic Algorithm**. This involves the application of optimization and service management principles in each level of implementation of the various building blocks in the optimized GUISET framework shown in fig 4.

Step 5: This involves simulating and performing appropriate comparative evaluations at three different levels:

Level 1: **GUISET Portal**- This is the job submission point for the end-user. It is multimodal in nature, however, job monitoring, grid resource information provisioning will begin at this point.

Level 2**: Deployment and Management**- is responsible for providing optimal job generation tool and service management tool for optimal results

Level 3**: Coordination and Resource Management** –This module is responsible for Grid resource allocation, coordination and services management.

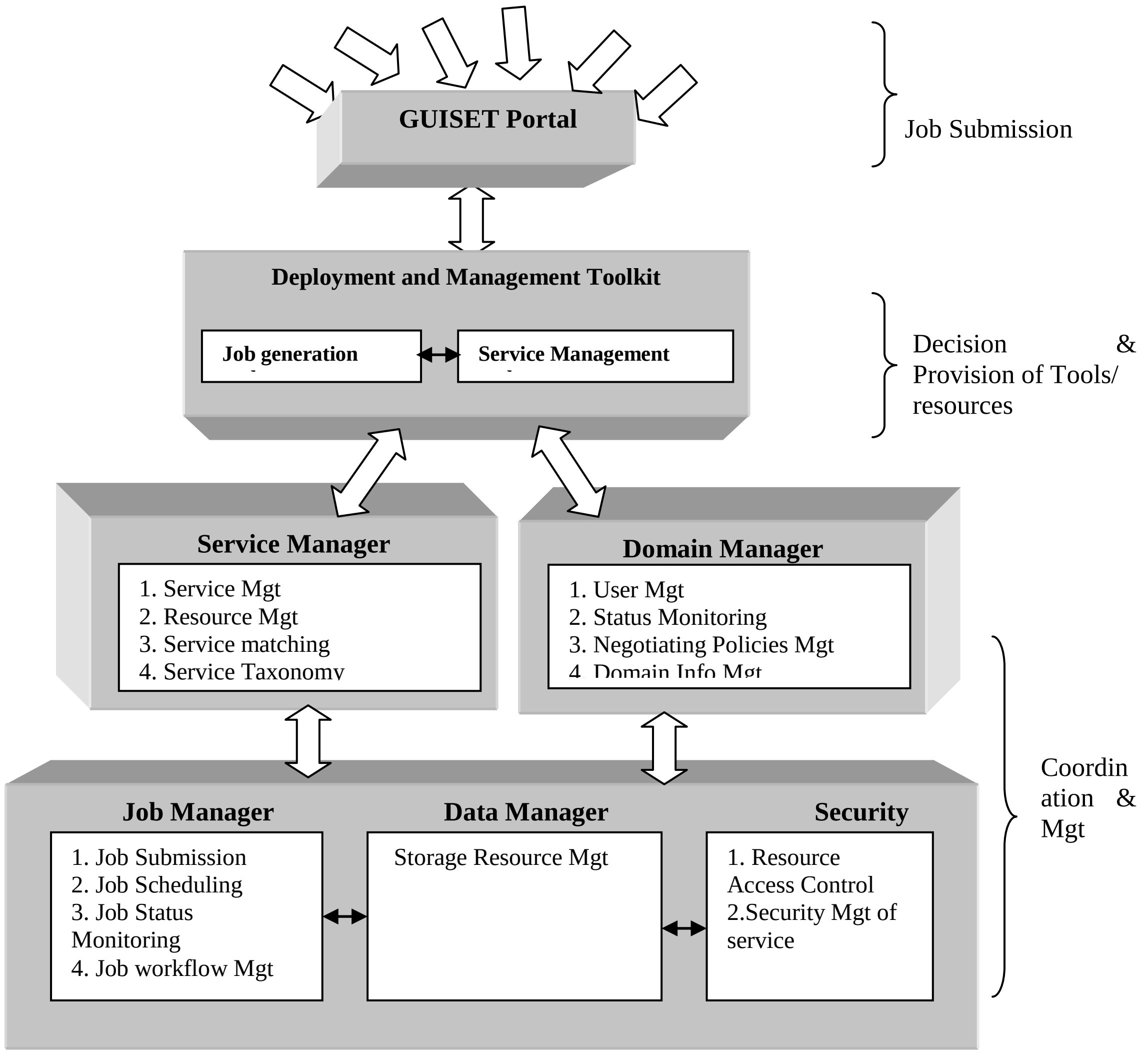


Fig 4. Building blocks for the optimal resource allocation and scheduling model of GUISET [20].

5. Conclusion

In this paper, we have proposed a neuro-dynamic approach for optimal scheduling and allocation of resources in a GUISET Enterprise Gird which is described as an NP-Complete problem. We formulated the resource allocation problem and propose a neuro-dynamic programming and service management solution approach for GUISET.

References

[1]. Ekabua, O. O. and M. O. Adigun: Experienced Report on Assessing and Evaluating Change Impact Analysis through a Framework and Associated Models**.** Journal of Information Science and Engineering. Vol. 25, No. 2, March 2009.

[2]. Burya, R., Abramson, D., and Giddy, J.: An Economy Driven Resources management Architecture for Global Computational Grids. The 7th International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2000), Las Vegas, USA, June 26-29, 2000.

[3]. Foster I, and Kesselman, C (eds) The Grid: Blueprint for a new computing infrastructure, San Francisco, CA Morgan Kaufamann(1998) http://www.mkp.compbooks-catalog/1-55860-475-8.asp

[4]. Foster I, Kesselman, C and Tuecke, S (2001) The anatomy of the Grid: Enabling scalable virtual organizations. International Journal of Supercomputer Applications 15(3) pp. 200 – 222,http://www.globus.org/research/papers.html1#anatomy

[5]. Hai Jin (2004) ChinaGrid: Making Grid Computing a Reality,Cluster and Grid Computing Lab, Huazhong University of Science and Technology, 430074, Wuhan, China.http://www.lib.sjtu.edu.cn/chinese/digital_library/caj_file/i_Jin-Hai-ChinaGrid-Jin.doc

[6].http://www.oracle.com/technology/tech/grid/index.html last accessed [01/07/06]

[7]. The Global Grid Forum, www.gridforum.org

[8]. Global Grid Forum, Proposed Scheduler Optimization Research Group, Scheduling and Resource Management Area, http://www.mcs.anl.gov/~jms/ggf-sched/WG/opt-wg.html

[9]. Global Grid Forum, Grid Resource Allocation Agreement Protocol Working Group, Scheduling Resource Management Area, http://people.man.ac.uk/~zzcgujm/GGF/graap-wg.html

[10].Huh E, et al (2000) Accommodating QoS prediction in an adaptive resource
management framework in "parallel and distributed processing" Rolim J. et al (eds) Lecture notes in computer science, Vol 1800, pp 792 – 799, Springer – Verlag, New York.

[11]. Fernandez-Baca D (1989), Allocating modules to processors in a distributed system,
IEEE Transaction on software engineering, Vol SE-15, pp. 1427 - 1436

[12]. Nemhauser G and Wolsey L (1999), Integer and Combinatorial Optimization, Wiley-Interscience, New York

[13]. Ali S. et al (2002) Greedy Heuristics for Resource Allocation in Dynamic Distributed Real-Time Heterogeneous Systems, in the proceedings of the International Conf. on Parallel and Distributed Processing Techniques and Applications (PSPTA '20), Las Vagas, Navaga

[14]. Maheswaran M. et al (1999), Dynamic mapping of a class of independent tasks onto heterogeneous computing systems, Journal of Parallel and Distributed Computing, Vol. 59, pp. 107-131.

[15]. Adigun M (2004) Software Infrastructure for e-commerce and e-business research working paper, Res-CSD-01 Centre for Mobile e-Services, University of Zululand.

[16]. Heymann, E., Senar, M., Luque, E. and Livny, M.: Adaptive Scheduling for Master-Worker Applications on the Computional Grid. Proceedings of the First IEEE/ACM International Workshop on Grid Computing, 2000.

[17]. Raman, R., Livny, M. and Solomon, M.: Matchmaking: Distributed Resource Mangement for High Throughput Computing, 7th IEEE International Symposium on High Performance Distributed Computing, 1998, pp. 28-31.

[18]. Casanova, H. and Dongarra, J.: NetSolve: Network enabled solvers, IEEE Computational Science and Engineering, 5(3) pp. 57-67, 1998.

[19]. Shao, G., Wolski, R. and Berman, F.: Performance Effects of Scheduling Strategies for Master/Slave Distributed Applications. Technical report TR-CS98-598, University of California, San Diego, September 1998.

[20]. Adigun, M. O.: Software Infrastructure for e-Commerce and e-Business research working paper, Res-CSD-01, Centre for Mobile e-Services for Development, University of Zululand, South Africa.

[21]. Tank D.W and Hopfield J.J (1987) Neural computation by time compression. Proceedings of National Academy of Science, USA, 84: 1896–1900

[22]. Garry J. (1987) *BYTES* ; Neural Network Heuristics. pp 183 – 191

[23].. Marbach P, Milhatsch O and Tsistsiklis J (2000), Call Admission Control and Routing in Integrated Services Networks using Neuro-Dynamic Programming, IEEE J. select. Areas Commun, Vol 18, no 2 pp 197-208.

Acknowledgements
This research is a contribution to an ongoing e-commerce software infrastructure project being developed at the Dept of Computer Science, University of Zululand. RSA and it is supported by Huawei-Telkom Centre of Excellence for Mobile e-services, NRF and THRIP.

Optimize Condor-G Matchmaker Service with IP Address

L. Mohammad Khanli[1], **H. Mohammadi**[2]
[1] Assistance Professor, Computer Science, University of Tabriz, Iran
[2] Islamic Azad University – Tabriz Branch, Tabriz, Iran

***Abstract** - The matchmaker mechanism is looking for the resource in base according to requester's orders, that contains details of the machine, for example CPU, memory and etc. In this paper, we add IP field to the matchmaker's database fields to increase speed of search in database and decrease time of search.*

Keywords: Condor-G, IP Address, Matchmaker, Grid

1 Introduction

Grid is a geographically distributed system that enables integration of heterogeneous resources needed for execution of complex, demanding scientific and engineering applications. Integration of resources beyond boundaries of a single organization also enables easier cooperation among diverse scientists who work in different geographical locations.

Grid middleware (GMW) is a set of services and protocols that enable seamless integration of resources in grid. It provides a layer of abstraction that hides differences in underlying technologies (e.g. computer clusters, storage managers, application services, etc.). Numerous standards are being defined for grid protocols and services, majority of them within Global Grid Forum (GGF) [1] organization. Basic functionalities of grid middleware are security, information, job and data management. Most widely used solutions that provide these basic functionalities are Globus Toolkit [2] and UNICORE [3]. Grid scheduling (also called super scheduling, meta scheduling and grid brokering) is one of the advanced features of grid middleware. It is defined as the process of scheduling jobs where resources are distributed over multiple administrative domains [4]. Up to date, there is no grid scheduling system that fully meets the requirements that will be described in following sections.

Execution of jobs in grid is shown in Figure 1.

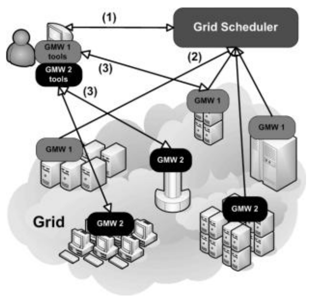


Fig. 1. Grid scheduling architecture

It consists of the following activities:

- Users submit their jobs described by using a description language to the grid scheduler (1),
- Scheduler uses grid middleware's information systems to discover and evaluate resources (2),
- Once the scheduler has defined where the job will be executed, execution is started and managed by using available grid middleware components (e.g. job and data management systems) (3).

Grid scheduling differentiates from classical cluster batch scheduling in many ways. In case of grid, a scheduler does not have full control over resources, information about resources is usually unreliable and stale, application behavior is difficult to predict. All these reasons make grid scheduling far more difficult to realize.

2 Condor and Condor-G Mechanism

The fundamental structure of the system has remained constant while its power and functionality has steadily grown. The core components are known as the *kernel*.

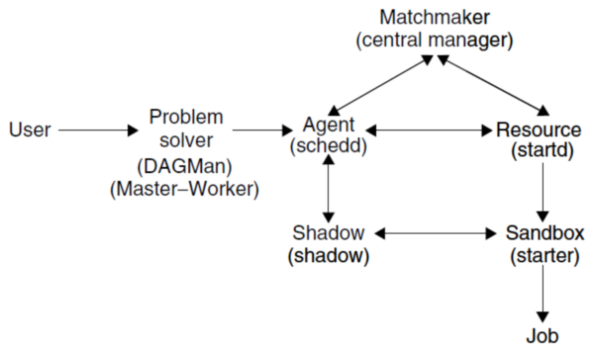


Fig. 2. The Condor Kernel

Briefly, the kernel works as follows: The user submits jobs to an *agent*. The agent is responsible for remembering jobs in persistent storage while finding *resources* willing to run them. Agents and resources advertise themselves to a *matchmaker*, which is responsible for introducing potentially compatible agents and resources. Once introduced, an agent is responsible for contacting a resource and verifying that the match is still valid. To actually execute a job, each side must start a new process. At the agent, a *shadow* is responsible for providing all of the details necessary to execute a job. At the resource, a *sandbox* is responsible for creating a safe execution environment for the job and protecting the resource from any mischief. Let us begin by examining how agents, resources, and matchmakers come together to form *Condor pools* [1].

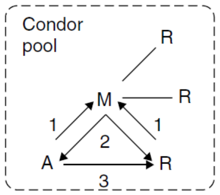


Fig. 3. A Condor pool

An agent (A) is shown executing a job on a resource (R) with the help of a matchmaker (M). Step 1: The agent and the resource advertise themselves to the matchmaker. Step 2: The matchmaker informs the two parties that they are potentially compatible. Step 3: The agent contacts the resource and executes a job. The agent makes the same information available to the community. A single machine typically runs both an agent and a resource daemon and is capable of submitting and executing jobs. However, agents and resources are logically distinct. A single machine may run either or both, reflecting the needs of its owner. Furthermore, a machine may run more than one instance of an agent. Each user sharing a single machine could, for instance, run its own personal agent. This functionality is enabled by the agent implementation, which does not use any fixed IP port numbers or require any super user privileges.

Each of the three parties – agents, resources, and matchmakers – is independent and individually responsible for enforcing their owner's policies. The agent enforces the submitting user's policies on what resources are trusted and suitable for running jobs. The resource enforces the machine owner's policies on what users are to be trusted and serviced. The matchmaker is responsible for enforcing community policies such as admission control.

As the Condor software developed, pools began to sprout up around the world. In the original design, it was very easy to accomplish resource sharing in the context of one community. A participant merely had to get in touch with a single matchmaker to consume or provide resources. However, a user could only participate in one community: that defined by a matchmaker. Users began to express their need to share across organizational boundaries.

3 Resource Ontologies

Resource ontologies are a critical component of the matchmaking framework. The ontology is an explicit specification of a conceptualization and a conceptualization is an abstract and simplified view of the world that we wish to represent for some purpose. The ontology defines a common structure that facilitates the sharing of information. It includes machine-interpretable definition of the basic concepts in a domain and their relations. Entities in the matchmaking framework, i.e. the providers, the requesters, and the matchmaking services, which are generally not in the same domain, must share the same ontology structure.

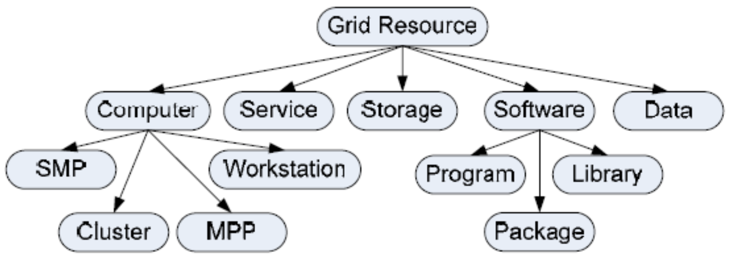


Fig. 4. The hierarchical relationship among grid resource classes.

The structure of a category of entities is described as a class in Protégé knowledge base [6]. Fig. 4 shows the hierarchical relationship among some grid resource classes. Fig. 5 shows the structures of the Workstation, Cluster, MPP, and SMP classes. A class consists of one or more slots. A slot describes one attribute of the class and consists of a name and a value. An instantiation of a class is called an instance of that class. The type of a slot value may be simple types, such as integer, float, Boolean, and string. For example, in Fig. 5, the value type of the slot NumberOfNodes of the class Cluster is an integer. The type of a slot value may also be an instance of a class. For example, in Fig. 5, the value type of the slot CPU of the class Cluster is an instance of the class CPU. A class may be extended from another class and inherit all the slots of that class. Fig. 6 shows the set of classes for software and hardware resources that are common in grid systems.

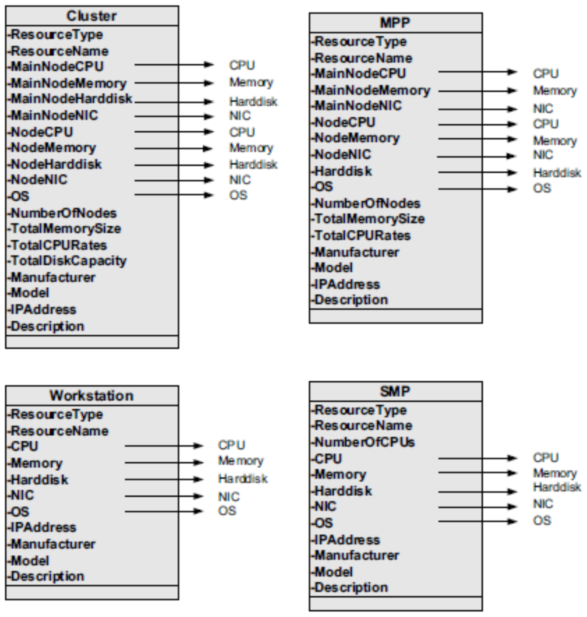


Fig. 5. The Workstation, Cluster, MPP, and SMP classes. The arrows to the right of some slots indicate that the values of these slots are instances of other classes.

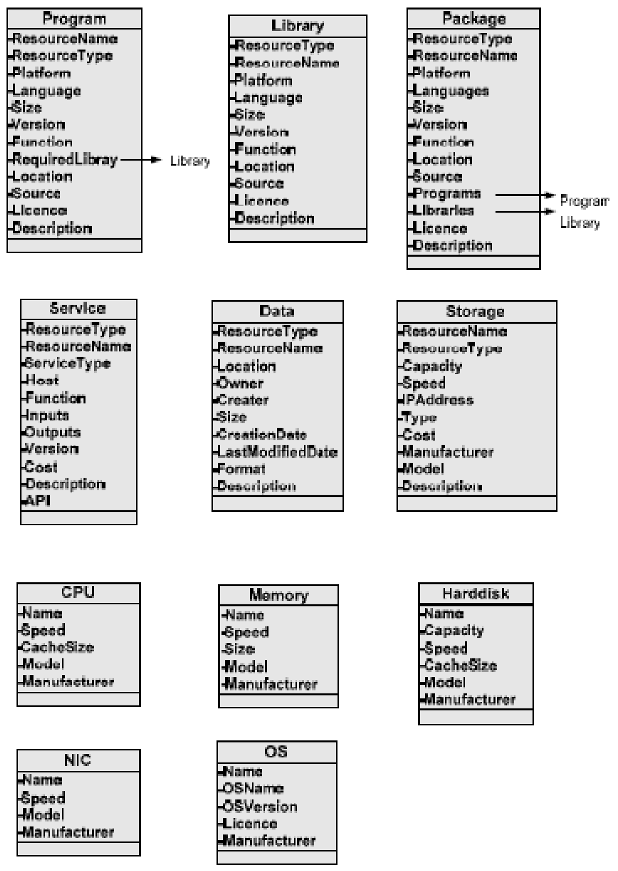


Fig. 6. Program, Library, and Package classes (top). Service, Data, and Storage classes (middle). CPU, Memory, Hard disk, NIC, and OS classes (bottom).

4 Searching for Resources

Grid Index Information Service (GIIS) is a Lightweight Directory Access Protocol (LDAP) server that collects the information related to every Grid Resource Information Service (GRIS) server available in the organization. This can be seen as a basic broker resource. It allows users or processes to find, if available, those computing resources and their particular features, such as memory and number and speed of processors.

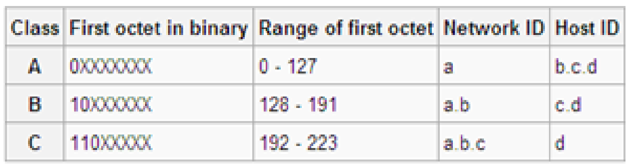
GRIS then acts as a super scheduler. Computational resources can be periodically queried about their features and status if a server is running Globus. This is because a GRIS is listening on a port. GRIS is a small LDAP server that stores static (and semi static) information, as well as dynamic information about available Grid hardware and the system software associated with it. GRIS can be looked at as an active index that allows for the simple central retrieval of a machine’s main features. GRB or any other resource broker can tap into this information and make use of it when looking at what resources to schedule for any given batch or real-time job.

5 IP Address

An Internet Protocol (IP) address is a numerical identification (logical address) that is assigned to devices participating in a computer network utilizing the Internet Protocol for communication between its nodes. Although IP addresses are stored as binary numbers, they are usually displayed in human-readable notations, such as 192.168.100.1 (for IPv4), and 2001:db8:0:1234:0:567:1:1 (for IPv6). The role of the IP address has been characterized as follows: "A name indicates what we seek. An address indicates where it is. A route indicates how to get there."

IPv4 uses 32-bit (4-byte) addresses, which limits the address space to 4,294,967,296 (232) possible unique addresses. However, IPv4 reserves some addresses for special purposes such as private networks (~18 million addresses) or multicast addresses (~270 million addresses). This reduces the number of addresses that can be allocated as public Internet addresses, and as the number of addresses available is consumed, an IPv4 address shortage appears to be inevitable in the long run. This limitation has helped stimulate the push towards IPv6, which is currently in the early stages of deployment and is currently the only offering to replace IPv4.

Classful network design allowed for a larger number of individual allocations. The first three bits of the most significant octet of an IP address came to imply the "class" of the address instead of just the network number and, depending on the class derived, the network designation was based on octet boundary segments of the entire address. The following table gives an overview of this system.



Class	First octet in binary	Range of first octet	Network ID	Host ID
A	0XXXXXXX	0 - 127	a	b.c.d
B	10XXXXXX	128 - 191	a.b	c.d
C	110XXXXX	192 - 223	a.b.c	d

Fig. 7. IP Classes

6 Optimize Matchmaker

IP Address like a postal code that shows address of connected machine to the network. Considering that, we can locate the local, city, state and country of the resource requester. Now a better idea for the matchmaker's search is, add a new field to matchmaker's database that contains IP address of resource. when a requester, order a resource matchmaker search that in his database according to the closer IP Rang to the requester's IP that to the classifieds of IP first looking in local so on search in city, state and country. So the first Premiership in matchmaker algorithm's search is find the closest IP to the requester's IP.

We know that each country have a particular IP range that in it we can separate each state, city and even local. For example: There is a requester is in Local A with 62.72.8.0 IP address that order a machine from a matchmaker, matchmaker first must search in requester's local in IP range from 62.72.8.1 to 62.72.8.255 then in requester's city that contains

IP range from 62.72.8.x to 62.72.10.x and then search in state and country and in the end search in the world for obtain the resource.

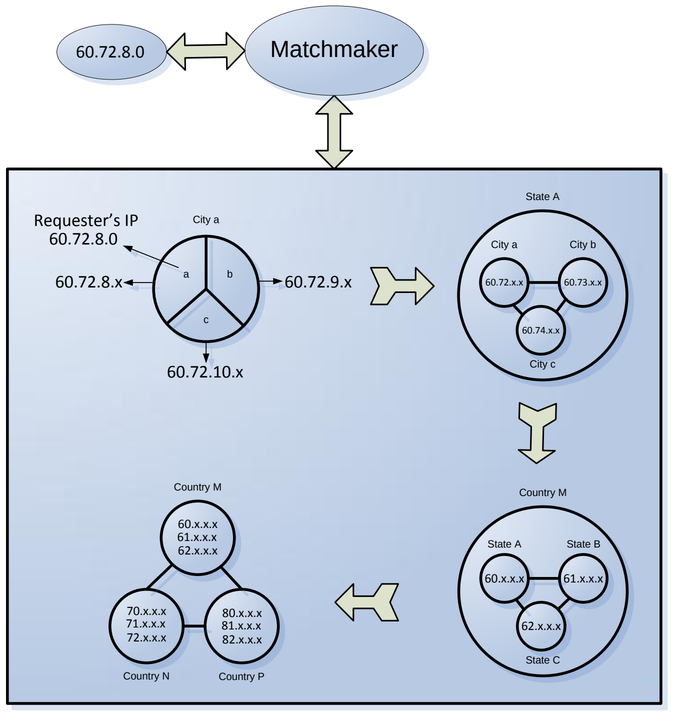


Fig. 8. New Algorithm

Assumption we have 10000 record in the bank for available resource and 1000 record is locate in the requester country that need the resource, for looking the requirement resource we first search it in the requester country and we search in only 1000 record.
If the requested resource finds our speed going up by 10 because we looking up in 1000 record and the records for search is less by 10 percent.

Uses the resource in the requester country has following ascendency:

- Use almost isotope topology and hardware.
- Speed of the internet in whole of country is almost isotope.
- The geographic distance between requester and resource is shorter than can have more security for connection.
- Time of search in database is minor and speed of search is higher.

7 Conclusions

In this paper we release a new table of match maker's database that this table contains of resource's IP for better use and rapid access to the resources.

8 References

[1] J. Frey, T. Tannenbaum, M. Livny, F. Foster, S. Tuecke, Condor-G: A Computation Management Agent for Multi-Institutional Grids, Cluster Computing 2002, 5(3), pp. 237–246.
[2] Globus Alliance, http://www.globus.org [02/28/2008].
[3] UNICORE, http://unicore.sourceforge.net/ [02/28/2008].
[4] J. NOVOTNY, S.TUECKE, V.WELCH, An Online Credential Repository for the Grid: MyProxy. In: IEEE International Symposium on High Performance Distributed Computing; 2001 Aug 7-9; San Francisco, USA. pp. 104–114.
[5] Douglas Thain, Todd Tannenbaum, and Miron Livny, "Condor and the Grid", in Fran Berman, Anthony J.G. Hey, Geoffrey Fox, editors, *Grid Computing: Making The Global Infrastructure a Reality*, John Wiley, 2003. ISBN: 0-470-85319-0
[6] ttp://protege.stanford.edu/
[7] April J. Wells : Grid database design, 2005 by Taylor & Francis Group
[8] RFC 760, "DOD Standard Internet Protocol". DARPA Request For Comments. Internet Engineering Task Force (January 1980).
[9] RFC 791, "Internet Protocol". DARPA Request For Comments 6. Internet Engineering Task Force (September 1981).
[10] Xin Bai, Han Yu, Yongchang Ji, Dan C. Marinescu: Resource Matching and a Matchmaking Service for an Intelligent Grid. International Conference on Computational Intelligence 2004: 262-265
[11] Emir Imamagic, Branimir Radic, Dobrisa Dobrenic : An Approach to Grid Scheduling by Using Condor-G Matchmaking Mechanism, Journal of Computing and Information Technology, Vol 14, No 4 (2006)

SESSION

GRID COMPUTING + CLOUD COMPUTING + INTERNET AND WEB COMPUTING + COMMUNICATION ISSUES

Chair(s)

TBA

Automatic Workflow Management System in Grid Web-Based OS Environment with Advanced Scheduling Module

Yi-Lun Pan [1], Chia-Yen Liu[1], Chang-Hsing Wu[1], Hsi-En Yu[1], Kuo-Yang Cheng[1] and Weicheng Huang[1]
[1]National Center for High-Performance Computing, Hsinchu, Taiwan
E-mail：serenapan@nchc.org.tw, chris@nchc.org.tw, hsing@nchc.org.tw, yun@nchc.org.tw, kuoyang@nchc.org.tw, whuang@nchc.org.tw

Abstract - *In a Grid Computing environment, there are various important issues, including information security, resource management, workflow management, routing, fault tolerance, and so on. Among these issues, the workflow management has emerged as one of the most important issues in past few years. Therefore, the research team of NCHC developed a workflow management system. The workflow management system can incorporate the properties of autonomic computing and exhibit the ability to reconfigure itself to the changes in the Grid environment. And, it can discover, diagnose, react to the disruptions of workflow execution, and monitor Grid resources automatically. And we also developed a workflow widget in Grid WebOS. The Grid WebOS is a web site that combined the Web-based Operating System (WebOS) platform with the Grid Computing environment to offer users a friendlier Grid environment. Currently, it joins the Grid, WebOS, and automatic resource selection mechanism to build a virtual computer in distributed environment. This progress helps to lower the barrier for using Grid Computing Environment. The research team of NCHC develops the Grid WebOS.*

Keywords: Widget, WebOS, Workflow, Grid Computing

1 Introduction

In the beginning of the 1990's, there was a brilliant success in Internet technology because of the birth of a new computing environment - Grid, which is composed by huge heterogeneous platforms, geographical distributed resources, and dynamic networked resources [1]. The infrastructure of Grid Computing is the intention to offer a seamless and unified access to geographical distributed resources connected via Internet. Thus, the facilities can be utilized more efficiently to help application scientists and engineers in performing their works, such as the so called "grand challenge problems". These distributed resources involved in the Grid environment are loosely coupled to form a virtual machine and each of these resources is managed by their own local authority independently, instead of centrally controlled. The ultimate target is to provide a mechanism such that once the users specify their requirement of resource. The virtual computing sites will allocate the most appropriate physical computing sites to carry out the execution of the application.

As a result, it has become necessary to provide Grid users with an interface that is both user-friendly and more straightforward. In this research, we combine the Web-based Operating System (WebOS) platform with the Grid Computing environment to offer users a friendlier Grid environment. By integrating Grid technologies – Globus Toolkit [2] and WebOS technologies, the National Center for High-performance Computing (NCHC) Grid development team has come up with a new and extremely lightweight approach to acquiring Grid services via Grid Widgets. As part of this research and to illustrate how useful our Grid Widgets are, we applied the technology to the development of a Laser-Plasma Simulation Widget and Obstructive Sleep Apnea Widget via automatic Workflow management system.

Besides, the current implementation also adopts the *Asynchronous JavaScript* and *XML* (AJAX) as part of the base of the WebOS. In order to enable and drive the core Grid middleware, the Globus Toolkit, the development team designed Grid Widgets with Java-PHP Bridge technology, which can improve the response and communication time of Grid Service. The major task of Grid WebOS and designed Grid Widgets can be easily customized and configured based on the end users needs. With these widgets, the Grid services implemented in this system then dynamically identifies the resources required by the end users with characterized resources. The jobs will then be submitted to local scheduler of the Grid resource selected and the job status will be monitored properly.

The NCHC Grid development team not only built Grid WebOS platform, along with the framework of EyeOS [3], but also incorporated self-developed Grid Widgets into the Grid WebOS platform. These Widgets are Resource Broker (RB) [4], Automatic Workflow, Information Retrieval, and so on. These Grid Widgets allow seamless and scalable access to Grid resources. This platform combines the Grid, WebOS, and automatic resource allocation mechanism to build a virtual computer in distributed computing environment. The technology of WebOS is the virtual operation system based on World Wide Web against traditional desktop systems. The utilization of the resources in heterogeneous Grid computing environment will be raised up because these Grid Widgets provide Grid users with an easy way of accessing integrated computing resources. In addition, an application-specific widget, such as Laser-Plasma Simulation Widget that integrates specific scientific research with distributed computing resources is implemented to ease the barrier of using the Grid computing environment for high-energy physicist.

Further, we also designed an advanced dynamic loading prediction-scheduling algorithm and implement it on scheduling module. It is developed according to each different required job criteria. The scheduling algorithm uses the dynamic loading prediction and adaptive resource allocation functions to meet users' requirements. The major task of the proposed multi-cluster resource manager is to dynamically identify and characterize the available resources, correctly monitor the queue status of local scheduler. Finally, the presented scheduling algorithm helps to select and allocate the most appropriate resources for each given job. The aim of the presented scheduling algorithm is to minimize the total time to delivery for the individual users' requirements and criteria.

The rest of the paper is organized as follows. Section 2 presents related works. In Sections 3, we propose workflow management system and describe the system architecture of Grid WebOS Platform. In Section 4, these Research Results of the Designed Grid Widgets are presented. Finally, the conclusion and future directions are presented in Section 5.

2 State of the Art

2.1 Existing Web-based Application Projects

When it comes to the graphical user interface (GUI) for Grid computing environment, GridSphere Portal Framework [5] is one of the most popular options. It is evolving to support a wide variety of applications that can be easily plugged into the portal by adopting the portlet API. With its success in the user management and portlet management, additional efforts related to the independent operation space, control thread, and applications are still to be tackled. Recently, there is a famous web application developed based on AJAX technique implementation, which is GMail by Google [6]. Its success in the file transfer and rich text editing illuminates that the development of Grid WebOS platform via AJAX technique become practicable. However, it does not provide on-demand requirements for applications, especially for Grid applications. Even though users can customize their personal homepage in Google, all the customizations are limited to the plain text and images.

Web-based Operating System (WebOS) project started at the University of California, Berkeley in 1996 as part of Network of Workstations [7]. So far, there are several typical commercial representatives of WebOS, such as YouOS [8], XIN [9], and so on. YouOS and XIN are online OS with Ajax and PHP techniques. However, these projects are not open source and also short of the management of distributed resources. To further enhance such an approach to meet the demands of Grid Computing, Grid Web-based Operating System (WebOS) platform, along with the framework of EyeOS has been developed. This development follows the spirit of open source, open standard and GNU/GPL license.

2.2 Existing Grid Workflow Projects

Over the last few years, a number of Grid Workflow Management systems such as Pegasus [10], Triana [11], Taverna [12], Condor DAGMan [13], Kepler [14], Gridbus [15] and Askalon [16] have been developed by projects around the globe. These systems focus on various aspects of workflow management including workflow expression language, graphical environment for workflow composition and execution monitoring, workflow scheduling heuristics, data management, legacy applications and fault-tolerant mechanisms.

Among these systems, Triana supports decentralized Peer-to-Peer (P2P) based workflow management. However, the P2P communication in Triana is implemented by JXTA protocol that uses broadcast technique. In our previous work [17], we use a DHT (such as Chord, Pastry, CAN) based P2P system for handling resource discovery and scheduling coordination. The employment of DHT gives the system the ability to perform deterministic discovery of resources and produce controllable number of messages in comparison to using JXTA.

With regards to Autonomic Computing paradigm, several research efforts have focused on enabling the autonomic properties into the system by addressing four main areas: self-healing, self-protection, self- configuration, and self-optimization. Projects in both industry and academia such as OceanStore [18], Storage Tank [19], have addressed autonomic behaviors at all levels such as hardware, software systems and applications. At the hardware level, systems may be dynamically upgradeable, while at the operating system level, active operating system code may be replaced dynamically. Moreover, at the application level, self- optimizing databases and web servers may be dynamically reconfigured to adapt service performance. In contrast, our work proposes to address these autonomic behaviors into the workflow management system.

2.3 Existing Resource Broker and Scheduling Algorithm

A general architecture of scheduling for resource broker which is defined as the process of making scheduling decisions involving resources over multiple administrative domains [20]. There are three important features, which are resource discovery, resource selection, and job execution from the previous research of a grid resource broker. As we know, several famous researches on resource broker are providing an integrated interface of accessing resources for different applications, such as Condor – G, EDG Resource Broker, AppLes, and so on [21], [22]. The resource brokers above also provide the capabilities of monitoring computing resources information and then allocate resources according to this information. Nevertheless, in some actual situations, the bottleneck of efficiency is not on the computing power but on the queues belonged to processors. Most of resources brokers as we know now did not take the information of real-time queuing status into consideration while making decisions of allocating resources, either to make precise and effective scheduling policy. On the scheduling algorithm scenario, the dynamic job scheduling is a crucial and fundamental issue in Grid Computing environment. The purpose of job scheduling mechanism is to find the optimal solution of resources allocation. A job scheduling strategy might not perform well in realistic environment if it allocates computing resources based on static information only, such as list heuristic and the listing scheduling (LS) [23]. Our presented research focuses on addressing this problem by means of implementing the algorithm contained the real-time queuing status calculation method additionally and integrating it to the Grid WebOS.

Furthermore, we design an automatic workflow management system since a finely designed algorithm usually will usually implement in a useful application or system. The automatic workflow system ingrates with the resources broker capable with new feature of mentioned above. It also helps scientists who need to run large-scale, complex-composed jobs in a multi-cluster or grid environments.

3 Proposed Grid WebOS & Workflow Widgets

3.1 Research Objective

By integrating Grid technologies and WebOS, we have come up with an approach to acquiring Grid services via Grid Widgets in Grid WebOS environment. The designed Grid WebOS has become necessary to provide Grid users with an interface that is both user-friendly and straightforward. In order to develop an autonomic workflow management system based on decentralized resource discovery architecture, we propose the workflow engine based on Grid WebOS. This research focuses on managing computing resources with an interactive graphical user environment.

3.2 Grid WebOS System Architecture

We develop several grid widgets for the purpose of providing a lightweight approach for users to acquire grid services that they wants to use. As the figure 1, when the NCHC Grid WebOS in the middle of this figure receives a grid job request from the users via the web browser, and then the job will be sent to the fittest cluster in the backend to process. The resource broker will help users to choose and allocate the resources and the workflow management system will help users to organize and submit their tasks using a graphical interface.

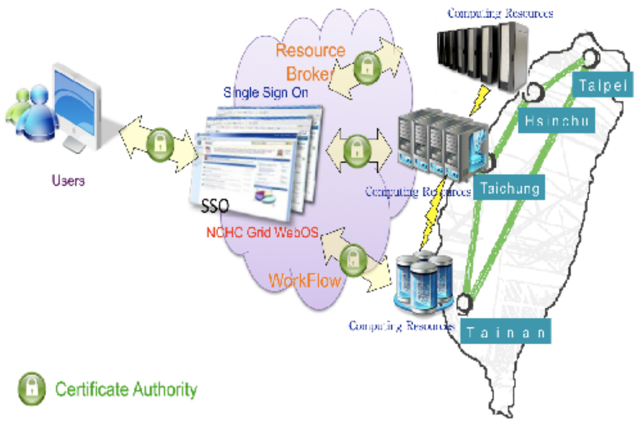


Figure 1. The System Architecture of Grid WebOS Platform

3.3 The Workflow Management System Architecture

In this section, we will explain the composing of our workflow engine including the functions of every component. As the figure 2 shows, users design their tasks and then constitute them to a single "workflow project" via the workflow widget on Grid WebOS. The workflow widget supports that users are allowed to customize the dependencies among tasks. In other words, users can decide the execution sequence of tasks. Furthermore, the system will also help users to decide if the output file of the source task is the input file of destination task while defining the dependency between two tasks. After finishing of defining the tasks and their dependencies, the workflow project will contain the descriptions of every single task in XML format or in other script format. After the "workflow project" submitting, then the Workflow Control Manager will create a thread to execute this "workflow project". Next step, the Request Module will receive the "workflow project" from the workflow control manager. It will manage the job submission with proper job attributes. The Request Module will convert the job's requirements into an object format and put the object into the Job Queue.

Following the Request Module, the Job Manager Module invokes the Resource Allocation Module to generate a best resource list. This resource list is based on the criteria posted by the job and the resource status. The Resource Information Module is responsible for providing the Resource Allocation Module with instantaneously resource status. The Information Module updates the objective of cluster resource status periodically. With the resources suggestion from the Resource Allocation Module, the job is then dispatched to the backend machines. The Job Manager Module receives the job from Resource Allocation Module. It will then send the job to the Execution Adapter to execute. As a consequence, the scientists can simply submit their “workflow project” to the Grid environment, and find the best physical computing site to dispatch and execute jobs. Thus, the users who even do not understand grid cluster very well, they can easily submit a series tasks and these tasks will be finished efficiently.

From users’ scenario, users just drag-drop their jobs with Workflow widget, and then it will generates a workflow specification. The designed Workflow engine can parse this workflow specification in Figure 3. Finally, the following Grid Middleware will dispatch jobs to the backend computing resources.

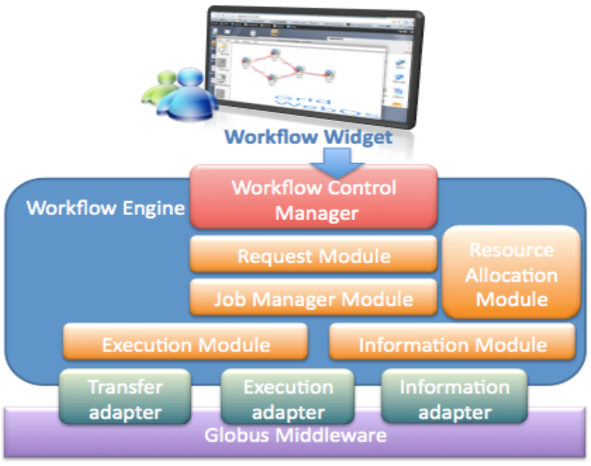


Figure 2. The Internal Architecture of Workflow Engine

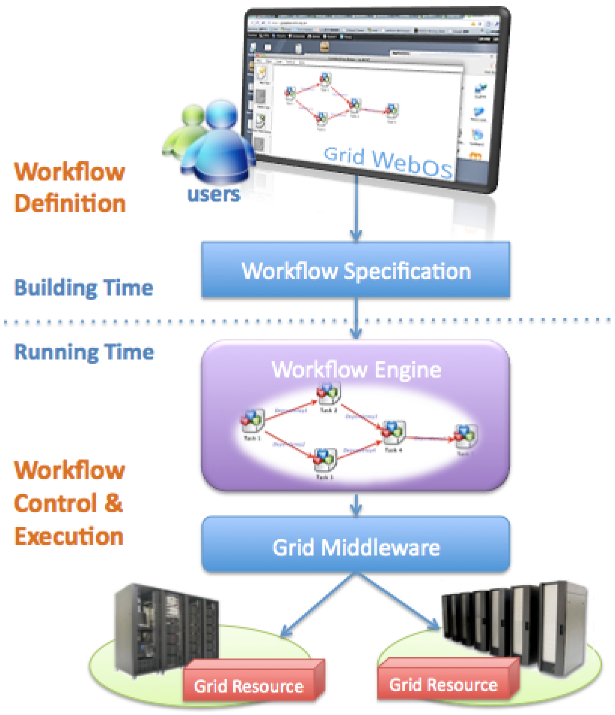
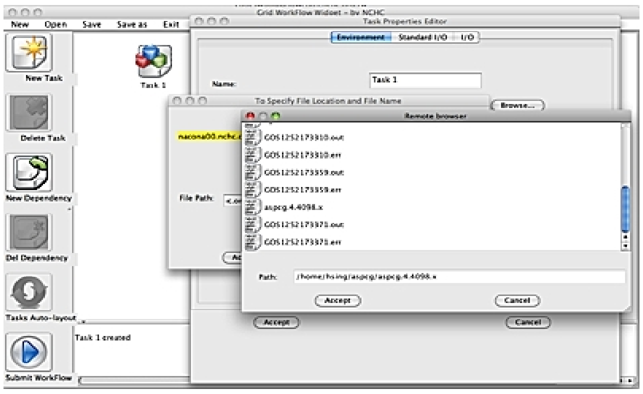


Figure 3. Workflow Management System

4 Research Results - The Designed Grid Widgets and Workflow Widget

In addition to the basic Grid Widgets, more advanced genetic computing widgets are attempted as well. One of the most important results in this paper is that we have developed many grid widgets with friendly graphical user interface, especially the workflow widget. All of these widgets can be easily used that users do not have to spend too much time learning it. Besides, the workflow widget is allowed users to customize and to arrange their complex tasks according to their requirements. These grid widgets for specific scientific applications are named the Workflow Widget, Resource Broker, Laser-Plasma Simulation Widget, and Obstructive Sleep Apnea (OSA) Widget.

The main function of the Workflow widget is to provide features of load prediction, adaptive resource selection, and characterization of the available resources, monitoring of the queue status of local scheduler, and automatic allocation of the Grid resource for job submission. All of these functionalities can be easily achieved with the drag-drop user interface. In figure4, users use Workflow Widget to create tasks and to define their attributes at first. Second, users then define the dependencies of tasks through workflow widget, as the figure 5 is illustrated.



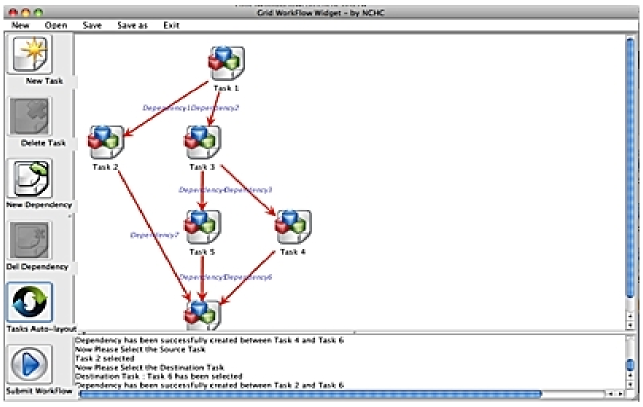
Name: Task 1

GOS1252173310.out
GOS1252173310.err
GOS1252173359.out
GOS1252173359.err
aspcg.4.4098.x
GOS1252173371.out
GOS1252173371.err

Path: /home/hsing/aspcg/aspcg.4.4098.x

Task 1 created

Figure 4. Workflow Widget – Create Tasks



```
Dependency has been successfully created between Task 4 and Task 6
Now Please Select the Source Task
Task 2 selected
Now Please Select the Destination Task
Destination Task : Task 6 has been selected
Dependency has been successfully created between Task 2 and Task 6
```

Figure 5. Workflow Widget – dependency of Tasks

As the figure 6 depicted, Resource Broker (RB) Widget provides the functionalities of load prediction, adaptive resource selection, and characterization of the available resources, monitor of the status queues of local scheduler, and automatic resource allocation which are integrated with the workflow widget.

The Laser-Plasma Simulation Widget provides specialized Grid services including Workflow to the high-energy physics community. It was used to solve relativistic high-harmonic generation problems formed when intense laser pulses interacted with an overdense (i.e. 1028m-3) plasma slab. The simulation domain was composed of 1500x500x500 cells that corresponded to a 12x12x18 m3 box. In the figure 7, it shows the snapshots of the solid target plasma density result. The surface of the solid target was vaporized and ionized until it became a layer of overdense plasma.

The Obstructive Sleep Apnea (OSA) Widget combines three-domain knowledge, including medical science, engendering, and Grid technology. Finally, the Grid users can read the 3D visualization, and even download the computed output via Grid WebOS directly, in Figure 8.

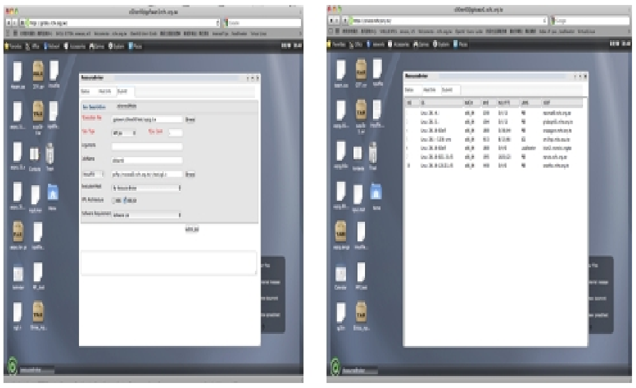


Figure 6. Resource Broker Widget

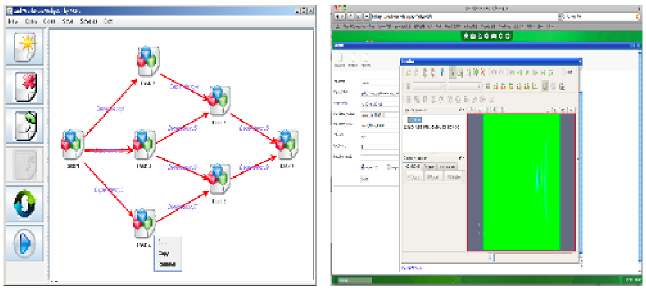


Figure 7. Customized Widget: Laser-Plasma Simulation Widget

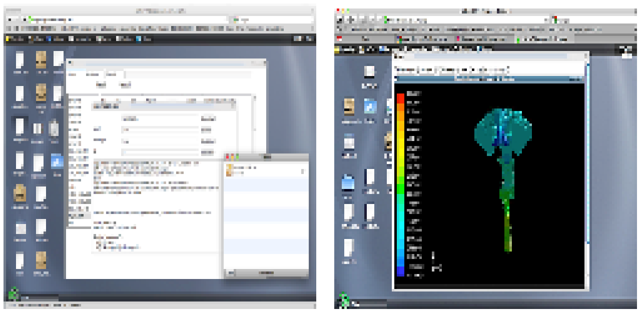


Figure 7. Customized Widget: Obstructive Sleep Apnea Widget

5 Conclusion and Future Work

The most important result in this paper is that we design and implement the integrated grid widgets on Grid WebOS. These grid widgets make users, especially scientists, who want to submit their complex tasks become easier and faster. The Grid WebOS and designed Grid Widgets can be customized and configured freely by the end users. It also possesses the abilities to dynamically identify, characterize available resources, and correctly monitor the queue status of local scheduler. These Grid Widgets are used to establish a virtual computing facility for the computing Grid services. The Grid Resource Broker and Workflow engine can automatically select the most appropriate physical computing resource, which pushes the Grid services a step further toward the world application. The Grid users do not have to manually select the computing resources any longer when they try to submit computing jobs. Our research can help grid users focus on developing their program without spending too much time learning and configuring the grids.

Such development is illustrated by a customized grid widget which focuses on the Grid service for the laser-plasma simulation via commercial package VORPAL©. With Workflow Widget, the physicists can submit their simulation job to the Grid environment simply by upload their input files and submit the job without worrying to where should the job be dispatched. In the future, other HPC-related researches such as Meta-computing via Grid environment and Grid-aware numerical algorithm will be investigated.

6 References

[1] R. AI-Khannak, and B. Bitzer, "Load Balancing for Distributed and Integrated Power Systems using Grid Computing," International Conference on Clean Electrical Power (ICCEP), 22-26 May, 2007, pp. 123-127.

[2] http://www.globus.org/

[3] http://eyeos.org/en/

[4] Yi-Lun Pan, Chang-Hsing Wu, and Weicheng Huang, "A Grid Resource Broker with Dynamic Loading Prediction Scheduling Algorithm in Grid Computing Environment," *the International Conference on Grid Computing Applications (GCA),* Las Vegas, NV, July 2008

[5] J. Novotny, M. Russell, and O. Wehrens, “GridSphere: a portal framework for building collaborations”, *IEEE Concurrency and Computations: Practice and Experience*, 16 (5), April 2004, pp.503-513.

[6] http://www.gmail.com

[7] Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau, David E. Culler, Joseph M. Hellerstein, and David A, Patterson, “Searching for the Sorting Record: Experiences in Tuning NOW-Sort,” *The 1998 Symposium on Parallel and Distributed Tools (SPDT '98),* Welches, Oregon, August 3-4, 1998.

[8] https://www.youos.com/

[9] http://www.xinteleport.com/default.asp

[10] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M. H. Su, K. Vahi, M. Livny, “Pegasus: Mapping Scientific Workflow onto the Grid”, *Across Grids Conference*, Cyprus, 2004.

[11] I. Taylor, M. Shields, and I. Wang, “Resource Management of Triana P2P Services”, *Grid Resource Management*, Netherlands, June 2003.

[12] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver and K. Glover, M.R. Pocock, A. Wipat, and P. Li, “Taverna: a tool for the composition and enactment of bioinformatics workflows”, Bioinformatics, 20(17): 3045-3054, Oxford University Press, UK, 2004.

[13] M. Litzkow, M. Livny, and M. Mutka, “Condor-A Hunter of Idle Workstations”, *In Proceedings of 8th International Conference of Distributed Computing Systems*, IEEE CS Press, USA, June 1988.

[14] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, and Y. Zhao, “Scientific Workflow Management and the KEPLER System”, *Concurrency and Computation: Practice & Experience*, Special Issue on Scientific Workflows, 2005.

[15] J. Yu and R. Buyya, “A novel architecture for realizing grid workflow using tuple spaces”, *In Proceedings of 5th IEEE/ACM Workshop on Grid Computing*, IEEE CS Press, USA, 2004.

[16] T. Fahringer et al., “ASKALON: a tool set for cluster and Grid computing”, *Concurrency and Computation: Practice and Experience*, 17:143-169, Wiley Inter- Science, 2005.

[17] R. Ranjan, M. Rahman, and R. Buyya, “A Decentralized and Cooperative Workflow Scheduling Algorithm”, *In Proceedings of 8th IEEE International Symposium on Cluster Computing and the Grid*, France, May 2008.

[18] J. Kubiatowicz, “OceanStore: Global-Scale Persistent Storage”, *Stanford Seminar Series*, Stanford University, Spring 2001.

[19] J. Menon, D. A. Pease, R. Rees, L. Duyanovich, and B. Hillsberg, “IBM Storage Tank–A Heterogeneous Scalable SAN file system”, *IBM Systems Journal*, 42(2):250–267, 2003.

[20] J. M. Alonso, V. Hernandez, and G. Molto, "Towards On-Demand Ubiquitous Metascheduling on Computational Grids," *the 15th Euromicro Conference on Parallel, Distributed and Network-based Processing (PDP)*, February 2007, pp 5.

[21] J. Schopf, "A General Architecture for Scheduling on the Grid, "*Journal of Parallel and Distributed Computing*, special issue, April 2002, p. 17.

[22] A. Othman, P. Dew, K. Djemame and I. Gourlay, "Toward an Interactive Grid Adaptive Resource Broker,"

Proceedings of the UK e-Science All Hands Meeting, Nottingham, UK, September 2003, pp. 4.

[23] M. Grajcar, "Strengths and Weakness of Genetic List Scheduling for Heterogeneous Systems," Application of Concurrency to System Design, 2001. *Proceedings. 2001 International Conference*, 25-29 June 2001, pp. 123-132.

Phantom Toolkit: A Grid-Enabled Implementation for Autonomic Cluster Computing in Computerized Classrooms

Shuen-Tai Wang, Chin-Hung Li, Chang-Hsing Wu, and Chih-Wei Hsieh
National Center for High-Performance Computing, Taiwan

Abstract - *Due to the successful achievement in current processor design and fabrication, the computing power of Personal Computer (PC) has become noticeable. Today, even the performance of individual laptop exceeds that of previous mainframe. Lots of high-throughput type of applications can be satisfied using desktop PCs like the ones found in computerized classroom. In this paper, we present a grid-enabled toolkit for cluster computing to utilize the computing power such as resides in computerized classrooms. The PCs in computerized classroom are usually setup for education and training purpose during the daytime, and shut down at night. After well development, these PCs can be transformed into a pre-configured cluster computing resource immediately without touching the existing education/training environment installed on these PCs. Thus, the training activities will not be affected by this additional activity to harvest idle computing cycles.*

Keywords: Personal Computer, cluster computing, computerized classroom

1 Introduction

Scientific computing has become the key player in advancements of modern science and technology. Over the years, the constant demand for computing power from "grand challenges" has resulted in the dramatic growth of supercomputing power. Also, the growth in computing power of the PC is a direct result of recent advances in computer science. Many high through-put types of applications can be satisfied by using the current desktop PCs, especially for those in computerized classrooms [1], thus, leaving supercomputers to perform the work of large-scale parallel computations.

Today, the computerized classroom is an unheeded resource or underutilized in academic institutes and colleges, but it is a potential computing resource to support some computations. The PCs in computerized classroom are usually setup during the daytime for education and training purposes and then shut down at night. One interesting topic can be raised is how to exploit the computing resource of the power-off PCs, and without touching the existing education/training environment installed on these PCs. Thus, the training activities will not be affected by this additional activity to harvest idle computing cycles.

In this paper, we propose a grid-enabled toolkit - "Phantom Toolkit" to solve the idling issue by deploying pre-configured cluster computing environment that can utilize the existing computing power residing in computerized classrooms. Phantom Toolkit was developed by National Center for High-performance Computing (NCHC) [2]. It includes two major components: Phantom Cluster and Phantom Grid. Phantom Cluster can transform the PCs into a cluster-computing resource immediately that can be used at night when the PCs are normally not in use without touching the existing education/training environment. This can be achieved by "Root over NFS" [3] diskless environment. So it does not touch the client hard drive, therefore, existing Operating Systems, along with all the software and applications installed on them, are preserved. In addition to automation and manageability, many middleware are packaged into Phantom Toolkit, such as resource management system, accounting system, monitoring tools, and the Message Passing Interface (MPI) [4] libraries for parallel program. Phantom Grid is responsible for collaborating the geographically distributed Phantom Clusters. We integrated some related grid middleware into Phantom Toolkit to perform reliable and efficient sharing of computing resources, and provide single entry interface for users for submitting, monitoring and controlling jobs in grid environment.

This paper presents a grid-enabled toolkit for autonomic cluster computing in wide-area computerized classrooms. It works well especially for computationally bound applications. The rest of this paper is organized as follows. Section 2 gives a description of hardware/software architecture. Section 3 gives some details of Phantom Cluster. Section 4 discusses the integration of grid middleware. Performance evaluation and analysis will be presented in section 5. Finally, section 6 presents the conclusion and future work.

2 System Architecture

Figure 1 shows the system architecture of Phantom. NCHC has three business units located at three science parks in Taiwan, and each business unit has two computerized classrooms. We built a Phantom Grid that consists of six Phantom Clusters which are widely distributed in computerized classrooms of NCHC over Taiwan. There is a front node in Phantom Grid. The front node is a login node located in the top of Phantom's architecture for users to provide a single entry interface for submitting, monitoring and controlling jobs. Again, there is a head node in Phantom Cluster. Each head node has two Ethernet interfaces. One is assigned a public IP address and connected to the front node via internet. The other is connected to the local classroom switch. Both front and head node are installed the same operating system to avoid the shared library files missing problem.

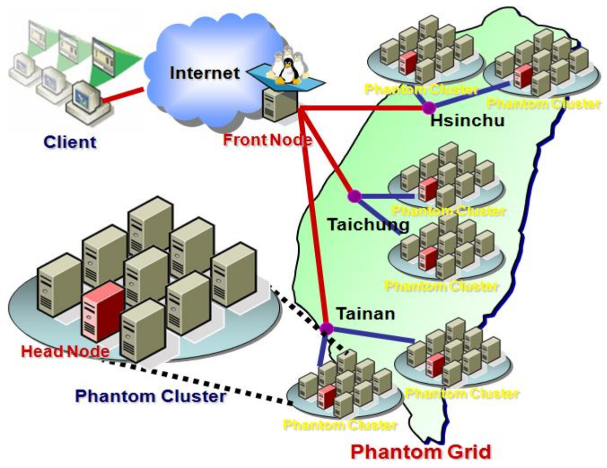


Figure 1. System Architecture

Figure 2 shows the software stack of Phantom Toolkit. It can be divided into two major parts: Diskless Module and Phantom Module. At the first, the Phantom Module depends on the proper setups of diskless environment. During the deployment of Diskless Module, the administrator needs to power on PCs one by one and gather their MAC addresses. The diskless module will collect MAC addresses and generate the associated configuration file for DHCP and PXELINUX [5] services. Furthermore, the list of MAC addresses will be the target of power management mentioned in the following section. After diskless environment being valid, the head node will become a powerful server, which provides initial RAM disk, IP address for PXE boot, NFS and NIS service for a computerized classroom. Above the diskless layer, many software are packaged into the Phantom Module for Phantom system, such as local queuing system, accounting software, monitoring tools, power management module, grid-enabled extension and MPI libraries. Most of the software are widely accepted and are tailored and tuned for helping automation and manageability. We also wrapped these pre-configured software into a package for some Linux distributions, including CentOS, Fedora, and Red Hat. With this one-size-fits-all solution staged, we can greatly eliminate Phantom system building efforts.

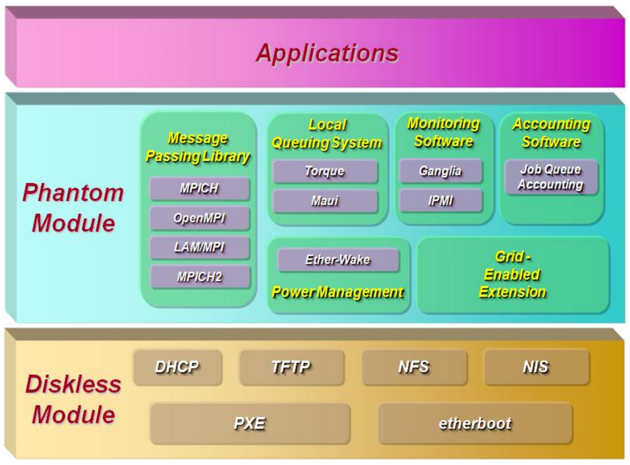


Figure 2. Software Stack

After well deployment, all PCs in the computerized classrooms could be scheduled. Users can login the front node and submit their jobs to resource broker anytime. The resource broker will fetch the jobs and send to the applicable Phantom Cluster by its scheduling policy. And then even when the work horses of the Phantom Cluster is not available at the moment. Submitted jobs will be queued and wait for computing resources to become available, typically during night time when the classroom PCs are free from their duty assigned to perform. When the PCs are available, the Phantom will fetch the suitable jobs, parse the requirements, and remote power on exact number of the PCs. After jobs are completed, the outputs will be sent back to the front node. While the Phantom with no further job request, it will power off the PCs that arranged by queuing system automatically. These features greatly reduce the management effort and time required to build a cluster as well as power consumption.

3 Phantom Cluster

Phantom Cluster is the base of Phantom system. It provides a pre-configured cluster-computing environment for utilizing the computing power resides in a computerized classroom. The special characteristics of Phantom Cluster as described in the following.

3.1 Power Saving

To echo today's energy saving issues, we also developed an approach to reduce energy utilization in Phantom Cluster. We do this work on the integration of resource management system and remote power management system [6] that aims at reducing power consumption such that they suffice for meeting the minimizing quality of service required by Phantom Cluster. In particular, our approach relies on recalling services dynamically onto appropriate amount of the PCs according to user's job request and temporarily shutting down the computers after finish in order to conserve energy. As shown in Figure 3, Phantom will wake up every minute to check job queue if there exist jobs, and make sure the PCs become available, Phantom then will fetch the applicable jobs, parses the requirements, and remotely powers on the correct number of PCs by Wake-on-LAN [7] protocol. After the job completes, Phantom powers the PCs down. Our implementation currently relies on checking the local queuing system (i.e. Torque [8,9]) job pool and then decides to shut down which compute nodes when no new job was submitted. By powering off idle PCs, it can significantly save more energy than always keeping all PCs running.

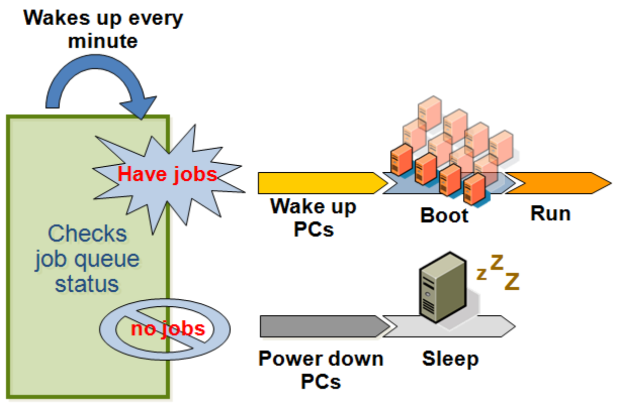


Figure 3. Scenario of Power Saving Mechanism

3.2 Classroom Management

In order to enable Phantom services work tightly with computerized classrooms in different locations. It is necessary to develop environmental controlling tools. In each computerized classroom in NCHC, it has already installed Clonezilla [10] for OS image cloning and deployment for education and training course. The Clonezilla has its own DHCP service to clone or backup OS images. But due to the DHCP protocol constrains, it is not allowed to have two DHCP servers under the same subnet at one time. Accordingly, we have to let the head node has the ability to start/stop the DHCP service of Clonezilla server automatically. The workflows of our classroom management are depicted in Figure 4 and Figure 5. On front node, we have arranged a central database storing the weekly education/training time table data. The Classroom-Ctrl tool client-side implementation on each head node will query this database every hour. This helps role of a classroom to change over time and facilitates our lecturers to give full-day or half-day courses. In other words, we could schedule the accessibility of each classroom.

Figure 5 illustrates the more detail workflow about how we control the DHCP service in a classroom. Before switching for cluster computing, the Classroom-Ctrl tool will shut down all PCs via UDP datagram, and stop the Clonezilla server's DHCP service. A few minutes later, Classroom-Ctrl tool will start head node's DHCP service to handle the boot-time installation requests from diskless clients. Each time slot for computing or training activity will last at least for two hours. In the same way, five minutes before the expiration on computing, the head node will power off its diskless nodes and stop its own DHCP service. The Classroom-Ctrl tool will try to start the Clonezilla's DHCP service again before time is up. With this workflow process, not only night time but also empty time slots, local PC classrooms will be ready for computing most of time.

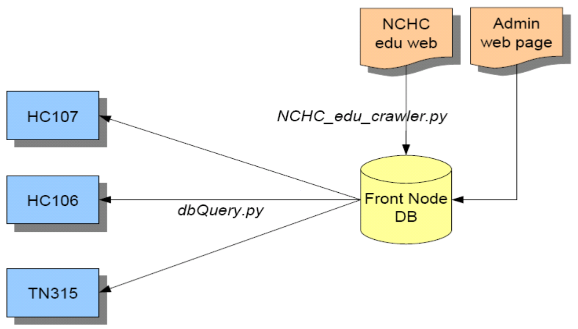


Figure 4. Front Node Workflow

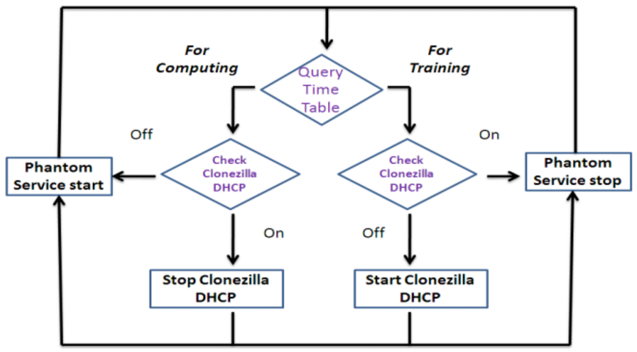


Figure 5. Head Node Workflow

4 Phantom in the Grid

After well deployment of Phantom Cluster in computerized classrooms, we hope that these distributed Phantom Clusters can collaborate and form as a whole resources for users. We refer to Grid architecture to achieve this work. Unlike the complexity of standard grid middleware, the Phantom's front and head nodes are all in our administrative domain, adopted the same operating system and user/group accounts. So we could bypass the credential management service in grid environment. For performing reliable and efficient sharing of computing resources between Phantom Clusters, we employ the meta-scheduler - GridWay [11,12] to provide a scheduling functionality.

GridWay is an open-source community project and it is highly modular, allowing adaptation to different infrastructures. We customized the prolog, wrapper, and epilog behavior in GridWay; moreover, we modified EM_MAD module to replace Globus [13] GRAM functions. Security is always the top priority for online services. We've replaced the GridFTP [14] with general SFTP/SCP service to migrate user data between front and head nodes. Installing GridWay makes central job submission and job dispatch to computerized classrooms in different geographical locations possible. Figure 6 shows how GridWay interacts with local cluster services. Hosting the GridWay meta-scheduler prevents us from installing additional software on remote head nodes. This also means than Phantom has the scalability for new head nodes to join it. We could add new computerized classrooms to raise Phantom's total computing capacity in the future.

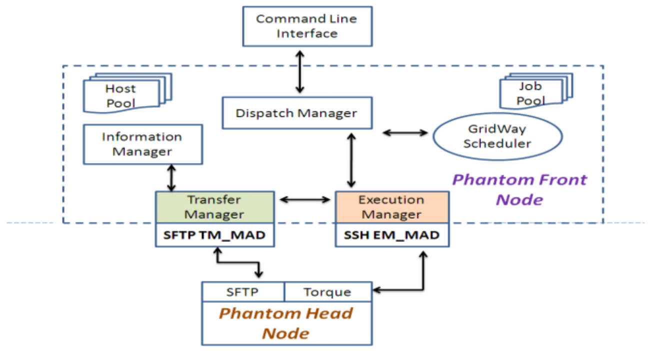


Figure 6. GridWay in Phantom

Like the way using a typical PC cluster, users need to prepare simple scripts before submitting jobs. Figure 7 lists a parallel job script example. There is no Queue Name specification in script any more. Instead, simply specifying the quantity of processors to NP variable is enough. The processor demand will be handled by a route type queue named "default" on the local cluster. Declaring this queue helps convey each job to the optimal execution one listed in Torque system.

GridWay job scripts will be transparently converted to equivalent Torque job scripts on head nodes. User's job files will be archived as a tar file and transferred to head node's working directory for execution when GridWay scheduler dispatched them. Users can specify their preferred classroom's head node IP address to the HOSTNAME variable in their scripts. If users did not overwrite the REQUIREMENTS variable, GridWay scheduler will choose one of computerized classrooms for each job by its round-robin ranking policy. Complicated Grid services sometimes hinder domain application scientists. For end users, GridWay provide a LRM-like (Local Resource Manager) console interface for submitting, monitoring and controlling jobs. Using Phantom system does not need to care the whole environment. These remote PCs will work as local resources connected to a cluster. Initially, Phantom Cluster was designed for temporary, small-scale computation. If there were some jobs remain running but the computerized classroom was running out of time, our customized scheduler will retain unfinished jobs' profile data on behalf of users. Then the Torque on the head node continues stopping all jobs to make the classroom available for education/training. These interrupted jobs will be automatically submitted again by our GridWay adaptation when the classroom is ready for computation next time.

```
#Job Name
NAME = meep_job
#Your Executable Program
EXECUTABLE = file:///work/james/job_meep/meep-mpi
ARGUMENTS = "./bend-flux.ctl"
INPUT_FILES = file:///work/james/job_meep/bend-flux.ctl
#Output Files
STDOUT_FILE = stdout.${JOB_ID}
STDERR_FILE = stderr.${JOB_ID}
#Setting Environment Variable
ENVIRONMENT =
MPI_V1_PATH=/opt/mpich/gnu,P4_GLOBMEMSIZE=99187896
REQUIREMENTS = HOSTNAME = "140.110.125.132"
#Job Type
TYPE = mpi
#Total 4 CPUs
NP = 4
```

Figure 7. Phantom Parallel Job Script Example

Currently, we have installed several scientific applications for online service, such as ABINIT [15], NAMD [16], Gromacs [17], Meep [18], etc. They were being successfully tested via the GridWay.

5 Performance Evaluation and Analysis

In this section, we employ HPL (High Performance Linpack) [19] to evaluate the performance of Phantom Cluster in a computerized classroom, and compare results with two PC clusters in NCHC. The two PC clusters are: the self-made cluster called Siraya [20] and the IBM 1350 Cluster. The details of hardware specification are listed in Table 1.

	Phantom	Siraya Cluster	IBM 1350 Cluster
CPU	Intel Core2 Duo 2.83GHz x 1	AMD Opteron 275 Dual Core 2.2GHz x 2	Intel Woodcrest 3.0GHz Dual-Core x 2
Memory	8G DDR2 SDRAM	8GB DDR400 Registered/ECC SDRAM	16 GB DDR2 PC2-5300 FBD ECC SDRAM
Network	Gigabit Ethernet x 1	Gigabit Ethernet x2, 10Gb/s InfiniBand	Gigabit Ethernet x2, 20Gb/s InfiniBand
Total Nodes	Desktop PC * 40	1U Rack-mount x 80	1U Rack-mount x 512
Watts	9.6 KW	66KW	238KW

Table.1 Hardware Specification of Three Clusters

HPL is to solve a dense linear system problem and is the widely used benchmark for evaluating performance of supercomputer systems. However, the performance of PC cluster is largely application-dependent. We also use the NCHC benchmark suite [21] which contains five benchmarks, namely hubksp, nonh3d, bem3d, ns3d and jcg3d, picked from four application domains. The hubksp program comes from physics, and nonh3d is an atmospheric science case. Both bem3d and ns3d are applications of parallel computing in the field of Computational Fluid Dynamics (CFD). The last one, jcg3d is from computational solid mechanics. These particular jobs helped us to distinguish the performance in different problem domains.

Table 2 shows the maximal performance (Rmax) and efficiency of various clusters on running 80 processors HPL job. We observe that the IBM cluster presents the best result among these machines, while Siraya is the worst one. Although this diskless based Phantom Cluster is not well-tuned on networking parameters, but it is capable of running parallel computations.

Cluster Name (NIC)	Network Topology	Switch	Rmax GFlops	Efficiency (%)
Siraya (IB)	Fat-tree	Voltaire 9288	271.05	77
Siraya (GbE)	Star	Nortel BayStack 5510-48T	228.8	65
Phantom (GbE)	Star	Cisco SLM2048	271.7	30
IBM Cluster 1350 (GbE)	Star	Force10 E600	652.8	68
IBM Cluster 1350 (IB)	Fat-tree	Voltaire 9288	758.4	79

Table 2. The Rmax and Efficiency of the Three Clusters

From sequential elapsed-time results listed in Table 3, Cluster 1350 excels Phantom while running single core except the jcg3d project. Phantom Cluster proved that it could provide capability for scientific computing. According to the floating point operations per clock cycle information, the Intel Core2 Duo doubles the AMD Opteron processor. For sequential jobs, Siraya apparently will be the worst one due to its less powerful processor. In terms of parallel processing, we evaluate hubksp and nonh3d jobs on many processors. Figure 8 shows the performance comparison of Siraya and Phantom. It is a comfort to see that Phantom took less time than Siraya while executing nonh3d. For parallel hubksp jobs, Siraya edged out Phantom on 32 cores. There is a plausible reason for results of Phantom while running the hubksp project beyond 16 cores. The main cause for a drop in performance is NFS network contention with the computing channel over the same switch in the diskless environment. Both Siraya and IBM Cluster 1350 have separate sub-networks for monitoring, computing and file-system to avoid conflict. Current desktop PCs are not dedicated for high performance computing. However, for non urgent jobs, Phantom will be qualified.

Considering the results in Figure 10, Phantom has better potential in performance-power ratio than Siraya, mainly because of cooperation of a group of inferior desktops. Generally speaking, one desktop consumes less power than a rack-mount server.

	bem3d	hubksp	jcg3d	nonh3d	ns3d
Phantom	2787	7562	2664	3598	9734
IBM Cluster	1240	1510	3765	1861	2252
Siraya	5690	11343	6121	8976	16335

Table 3. Execution Time (sec.) of NCHC Benchmark Suit

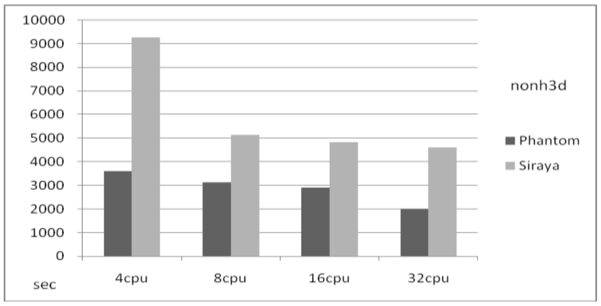


Figure 8. NCHC Benchmark Suit – nonh3d

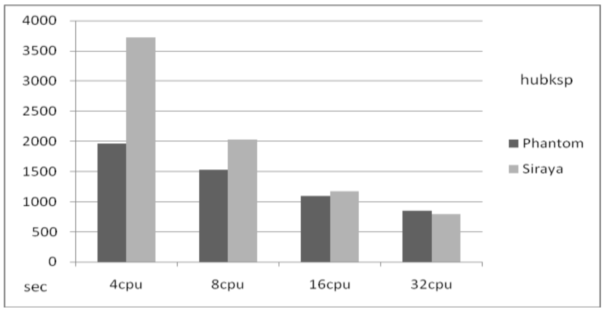


Figure 9. NCHC Benchmark Suit – hubksp

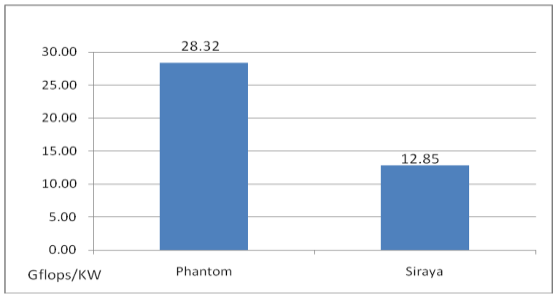


Figure 10. Performance/Power Ratio Comparison

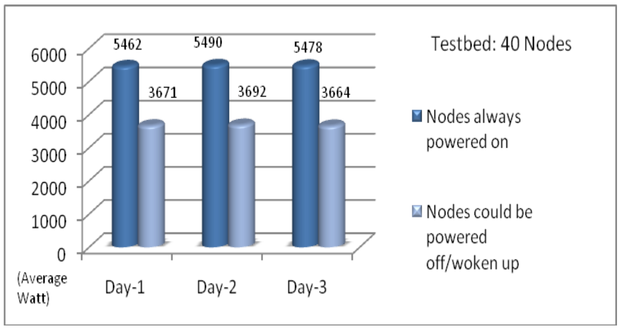


Figure 11. Power Consumption Comparison

Figure 11 shows we can reduce power consumption while Phantom's power saving mechanism is launched. The first case is to keep PCs powered on all the time. The other case is that PCs were powered off when no requests. However, running other applications or different submission frequencies will cause distinct results, so we submitted the same workload (240 HPL jobs; 10 jobs per hour) in three days for testing cases to sustain objectivity. Using power saving mechanism roughly saved 1790 watts per hour compared to the always powered-on case. If we keep running these two cases for months, the distinction will be more obvious.

Phantom Cluster takes advantage of existing idle time of computerized classrooms to do computing. It makes sense to neglect the purchasing cost. Besides, the power consumption of booting stage will obviously be reduced in diskless mode. Running Phantom will be more economical than staging a traditional PC cluster. We can reduce electricity and disk failure cost for online service. To sum up, C-P ratio of Phantom Cluster will far surpasses installing brand new machines.

6 Conclusion and Future Work

Phantom utilizes the untapped computing power that resides in computerized classroom PCs. It works especially for sequential or parallel computationally bound applications very well. But it does not apply to communication bound applications which need high-performance file system or local scratch space on compute nodes. To echo today's energy saving issues, Phantom toolkit includes a built-in resource allocation mechanism that wakes up the idle PCs when computing power is needed and then powers them down when the job is done. The Grid middleware are also integrated into Phantom Toolkit to perform reliable and efficient sharing of computing resource resides in geographically distributed computerized classrooms. For specific science applications, the benchmark results indicate that Phantom not only has the advantage of better cost/performance ratio, but also has the potential to outperform high-end machines. Phantom Toolkit also greatly reduces the time required to build and manage a cluster. In the future, we plan to add Cloud middleware that Phantom can be extended for Cloud Computing. It also will soon be planned to deliver freely as one CD/DVD to provide a Drop & Go cluster/grid computing solution.

7 References

[1] A. Apon, R. Buyya, H. Jin, and J. Mache, "Cluster Computing in the Classroom: Topics, Guidelines, and Experiences"
[2] NCHC, National Center for High-performance Computing, http://www.nchc.org.tw
[3] C.T. Yang, P.I. Chen, and Y.L. Chen, "Performance Evaluation of SLIM and DRBL Diskless PC Clusters on Fedora Core 3," In Parallel and Distributed Computing, Applications and Technologies, 2005.
[4] Message Passing Interface Forum, "MPI: A message-passing interface standard," International Journal of Supercomputer Applications 8 (3/4) (1994) 165-414
[5] PXELINUX, Available: http://syslinux.zytor.com/wiki/index.php/PXELINUX
[6] A. DiRienzo, John A. Medeiros, M. Whitlock, E. Wages, and J. Highfield, "Concepts for Computer Center Power Management," International Supercomputing Conference, 2010.
[7] Wake-on-LAN, Available: http://en.wikipedia.org/wiki/Wake-on-LAN
[8] TORQUE Resource Manager, Available: http://www.clusterresources.com/products/torque-resource-manager.php
[9] K. Chadalvada, S. Nidoni, and T. Sebastian, "Efficient Power Utilization of a Cluster Using Scheduler Queues," Available at: http://www.hipc.org/hipc2006/posters/cluster.pdf
[10] S. Shiau, "The Clonezilla project," Available: http://clonezilla.nchc.org.tw and http://clonezilla.sourceforge.net
[11] H. Eduardo, M. Ruben, L. Ignacio M., "An Experimental Framework For Executing Applications in Dynamic Grid Environments," ICASE Technical Report (2002)
[12] P. Armstrong, "Building a Scheduler Adapter for the GridWay Metascheduler," Faculty of Engineering Summer 2006 Work Term Report
[13] I. Foster and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit," International Journal of Supercomputer Applications 11 2 (1997), pp. 115-128.
[14] The Globus Project: the GridFTP protocol, Available: http://www.globus.org/datagrid/gridftp.html
[15] ABINIT, Available: http://www.abinit.org/
[16] NAMD - Scalable Molecular Dynamics, Available: http://www.ks.uiuc.edu/Research/namd/
[17] Gromacs, GROningen MAchine for Chemical Simulations, Available: http://www.gromacs.org/
[18] MEEP, Available: http://ab-initio.mit.edu/wiki/index.php/Meep
[19] A. Petitet, R. C. Whaley, J. J. Dongarra, and A. Cleary. "HPL - A Portable Implementation of the High-performance Linpack Benchmark for Distributed Memory Computers," Available: http://www.netlib.org/benchmark/hpl/
[20] Siraya Cluster, Available: http://siraya.sro.nchc.org.tw/
[21] K.C. Huang, H.Y. Chang, C.Y. Shen, C.Y. Chou, S.C. Tcheng, "Benchmarking and Performance Evaluation of NCHC PC Cluster, "The Fourth International Conference on High-Performance Computing in the Asia-Pacific Region-Volume 2, 2000

High Performance Cloud Computing: An Emerging Strategy for Scientific Computing

E. Okorafor
Computer Science, African University of Science & Technology, Abuja, Nigeria

Abstract - *Scientific computing often requires the availability of a massive number of computers for performing large scale experiments. Traditionally, high-performance computing solutions and installed facilities such as clusters and super computers have been employed to address these needs. Cloud computing provides scientists with a completely new model of utilizing the computing infrastructure.*

The infrastructure services (Infrastructure-as-a-service), provided by these cloud vendors, allow any user to provision a large number of compute instances. However, scientific computing is typically characterized by complex communication patterns and requires optimized runtimes. This paper will outline the challenges in exploiting cloud computing infrastructure and technologies for scientific computing, and its potential applications, especially, in the context of parallel applications, system performance, utilization, overheads and runtimes.

Keywords: Cloud, grid, virtualization, high-performance computing

1 Introduction

The unprecedented introduction of the commercial cloud infrastructure and services have allowed users to provision compute resources, storage resources, as well as applications in a dynamic manner on a pay per use basis. These resources are relinquished when not used and can be integrated within an existing infrastructure. Some of these infrastructures and services such as Amazon EC2/S3[1,2], Gogrid[3], Microsoft Azure [4], Google App Engine [5] and others, have the effect of improving resource utilization, decreasing waste, and improving energy conservation. In addition solutions employing cloud technologies such as MapReduce[6], Hadoop[7] and Dryad [8] have proved to be successful.

The resources provisioned are transparent to the users. In some cases, the use of virtualized resources actually give the user much more control over the provisioned resources, including the ability to completely customize the Virtual Machine (VM) images, root/administrative access, etc. A combination of open-source cloud infrastructure software such as Nimbus [9], Eucalyptus [10] and open source virtualization software such as Xen Hypervisor [11], allow organizations to supplement their existing computation facilities by leasing from commercial cloud infrastructure, while improving resource utilization, as in the case of private clouds.

With all these, an assumption can be made that access to computational power is no longer a barrier to scientific computation which usually requires the availability of a massive number of computers performing large scale data/compute intensive applications. Scientific computation typically involves the construction of mathematical models and numerical solution techniques to solve scientific, social and engineering problems. Many of these problems tend to be large and complex in scope that dedicated high-performance computing (HPC) infrastructures such as grids, clusters or network of workstations have traditionally been utilized to address these computing needs.

1.1 HPC clusters, grid and cloud computing

Grid and cluster computing have opened up many opportunities and created profound capabilities for innovative research with an extraordinary advantage of on-demand computing power as a utility much like electric power. These capabilities include dynamic resource or service discovery, and the ability to pool a large number infrastructural resources belonging to different administrative domains. Additionally, the intrinsic ability to find the best set of compute and storage nodes to accomplish a specific scientific computational task is at core of the success of these HPC clusters and grid computing. Some of these grids for scientific computing [12] have become so successful that a world-wide infrastructure has been established and now available for computational science.

With all the above promising features of the cloud, HPC has not been a good candidate for cloud computing due to its requirement for tight integration between compute nodes via low-latency interconnects. In addition, virtualization, which is often prerequisite for migrating local applications to the cloud, does not allow for scalability and efficiency in an HPC context due, to its inherent performance overheads. HPC clusters usually run fully-utilized and therefore there are no benefits gained from consolidation. The primary focus of cloud computing has been geared towards non mission-critical or non-performance-demanding applications.

1.2 Challenges and motivation

In the context of scientific computation using HPC these three major preconditions need to be satisfied:

1. There need for tight coupling between the compute nodes via low-latency interconnects
2. The application needs to be parallizable to utilize the available resources
3. There should be an appropriate parallel runtime to implement it

Therefore, in order to migrate scientific applications to the cloud, the aforementioned preconditions need to be satisfied. From the research point of view, initial studies have been conducted on the feasibility of using the cloud and/or cloud technologies for scientific computing. Some research have focused on using cloud computing technologies by analyzing the performances of HPC scientific applications [13] or the cost of performing scientific experiments [14] on cloud infrastructure. Different solutions are potentially available to move from the traditional science grids and HPC clusters and embrace cloud computing paradigm. Some vendors employ hardware level virtualization to provide compute and storage resources on demand. Another solution provided by vendors focus on application level virtualization by enforcing a specific application model that leverage their large infrastructure and scale up and down on demand. Other solutions provide end users with a platform for developing cloud computing applications, resulting in a better Quality of Service to the end user.

In this paper, we explore and assess the performance, overhead, system utilization and runtimes of HPC applications in the cloud environments. The paper also addresses the need for virtualization in HPC clouds. We conclude that HPC can use the cloud concept without an appreciable decrease in performance, especially in the case of a private cloud if virtualization is excluded. Achieving native application performance and scalability in cloud environments will enable HPC to effectively extend itself in such environments. This will certainly make it easier for research centers and universities to utilize high performance compute resources across the world for better research, education and product development.

The rest of the paper is organized as follows: first we provide an overview of HPC environment and its role in scientific computing. Next, we present cloud computing by defining the reference model and the key elements in this paradigm, followed by the discussion of the feasibility of using cloud infrastructure for HPC. In section 5, we discuss the performance issues related to HPC in the cloud and implications of using virtualized resources. Final thoughts and key observations about the future directions of cloud computing, as a valid support for scientific computing, are discussed at the end.

2 High performance computing environments

HPC in scientific computing helps to accelerate speed or reduce the time to obtain computational results. It provides significant cost reductions and tremendous flexibility. At the core strength is the ability to drive the CPU performance towards its limits. The trend over the past decade has seen HPC migrate from supercomputers to commodity clusters. The drivers for this have been the desire to achieve high performance at lower cost (best cost/performance).

2.1 Node Architecture

The HPC architecture has a huge influence on the performance. For example to meet the demands of more powerful HPC nodes, more execution cores (e.g. dual- and quad-cores) are being integrated into each processor and more processors are being tightly connected. The HPC architecture is characterized by packet-switched, high-bandwidth, scalable, low-latency point-to-point interconnection technologies. There are important challenges in this strategy to keep the power consumption as low as possible while increasing the computational capabilities of the HPC nodes.

2.2 Cluster Interconnect

The networking requirements of each node must match that of the CPU so that the application can scale accordingly, without introducing additional network or CPU overheads. The InfiniBand has emerged as the most deployed high-speed interconnect because it provides low-latency, high-bandwidth and extremely low CPU overhead.

2.3 Parallel Processing

Traditionally, software programs run on HPC clusters are parallel applications using message passing infrastructures such as PVM and MPI to achieve fine grained parallelism. Applications achieve coarse-grained parallelism using workflow frameworks. The applications can then be scheduled using software systems such as Platform LSF HPC. In such an environment the availability of computational power becomes the focus of resource allocation. The application and data required needs to be moved to the available computational power. The efficiency of the overall system may decrease due to data movement through the network, highlighting the need for low-latency, high bandwidth network infrastructure.

3 The rise of the clouds

Cloud computing is a term used to describe the infrastructure, platform and type of application. A cloud computing environment encompasses the ability to dynamically provision, configure, reconfigure and de-configure servers and services as needed. The servers or nodes can be physical or virtual machines. The term cloud computing is too broad to be captured into a single definition. The key elements include software and hardware available on demand over the internet.

3.1 Definition and services

Buyya et al. [15] gives a more structured definition as a *"type of parallel and distributed system consisting of a collection of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreement"*. This definition is apt in that it adequately captures the ability of delivering both infrastructure and software as services. The resources that constitute the physical infrastructure include clusters, datacenters, servers or desktop machines. The services exposed by the cloud can be classified into three major offerings available to the end-user. These are; 1) Infrastructure-as-a-Service (IaaS), 2) Platform-as-a-Service (PaaS), and 3) Software-as-a-Service (SaaS)

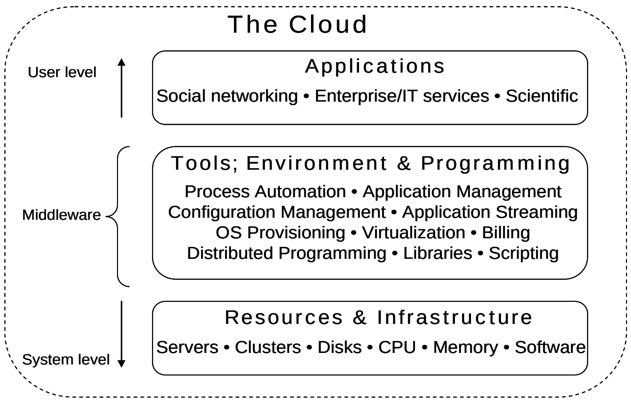

The Cloud
User level
Applications
Social networking • Enterprise/IT services • Scientific
Middleware
Tools; Environment & Programming
Process Automation • Application Management
Configuration Management • Application Streaming
OS Provisioning • Virtualization • Billing
Distributed Programming • Libraries • Scripting
System level
Resources & Infrastructure
Servers • Clusters • Disks • CPU • Memory • Software

Figure 1: Cloud computing layered architecture and components

Clouds can be internal (private), managed with an organization, providing compute resources to the organization's employees. When these resources are managed by a cloud computing vendor, then it is external (public). Figure 1 shows the complete structure of the cloud, listing the different layers and components.

3.2 Cloud for HPC

It is then conceivable that clouds can be built with the HPC systems' components, and if this incorporates a low-latency networking solution, then it is conceivable that the cloud can provide HPC-as-a-Servive (HPCaaS), or as high performance cloud computing. The main challenge with using the cloud for HPC is the performance or productivity. As earlier mentioned, HPC applications or scientific computations require high compute power, high performance interconnects and fast connectivity to the storage or file systems. Most clouds are built around non-homogeneous distributed systems, connected by low-performance interconnects. These solutions do not present optimal environments for HPC applications or scientific computing. In addition, virtualization, which enhances the provisioning process, increases the load on the compute infrastructure and further degrades cloud performance.

4 High performance cloud computing: Evaluation & analysis

Following earlier sections, we can summarize that in order to use cloud infrastructure for high performance computing, the cloud needs to meet the following requirements;

1. design based on HPC components
2. with high performance, low latency, high throughput interconnects
3. parallizable applications to utilize resources
4. appropriate parallel runtime to implement it.

In order to determine the feasibility, an HPC cloud environment was built as described in Section III. The system configuration is summarized in Table 1.

Table 1 – System configuration

Servers	HP ProLiant ML115 G5 64-node cluster
Processors	2 Dual core 2.2GHz AMD Opteron processors per node
Memory	4 x 2GB, 667MHzRegistered DDR-2 DIMMs per node
OS	CentOS 5.2
Interconnects	Mellanox MT25408 ConnectX DDR InfiniBand OpenFabrics Enterprise Distribution (OFED) 1.4 software stack
Application	MPI (Matrix Multiplication)
Scheduler	Platform LSF HPC v7
Workload Monitor	Platform RTM v2
Cluster Management	Condor

For our preliminary evaluations, we measure the performance of the cloud computing technologies; MPI, MapReduce and Hadoop applications on the cluster and then calculate the overhead induced by the different parallel runtimes using the following formula;

$$k = (p * t_{(p)} - t_{(1)}) / t_{(1)} \qquad (1)$$

Where p denotes the number of parallel processes and t is the time as a function of the number of parallel processors used. $t_{(1)}$ is the time taken for execution on a single process. The result of this analysis is shown in Figures 2. From the result, we can observe that the cloud runtimes show considerable overheads compared to the MPI versions of the same Matrix multiplication application. This means that for iterative and complex classes of applications, high performance parallel runtimes are definitely needed. Parallel applications implemented using message passing runtimes can utilize various communication constructs to build diverse communication topologies. Typical cloud runtimes are based on data flow models, and as such, lack the ability to support this complex communication constructs. However, we expect that the cloud runtimes will perform competitively well for task parallel and MapReduce style applications.

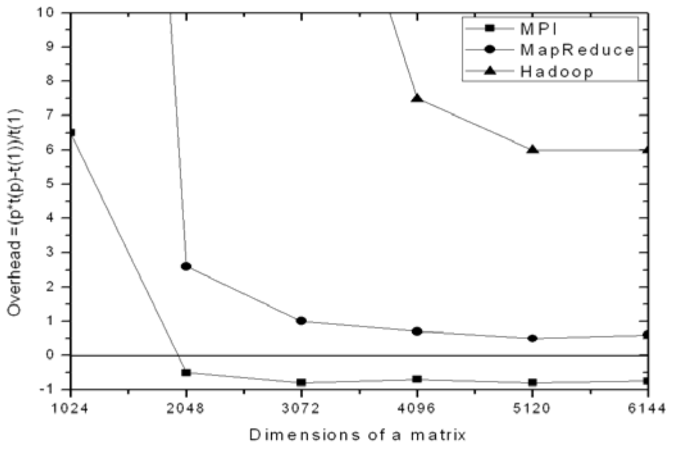


Figure 2: Overhead incurred by the different programming runtime for the matrix multiplication application

5 Results & performance issues

Following the previous observation, we analyzed the performance implications of cloud computing infrastructure for parallel applications implemented using MPI as compared with the traditional HPC clusters. Specifically, the overhead of virtualized resources, computation and communication latencies of the applications on cloud resources are of interest. The infrastructure setup we used is described as follows:

1. Native (Bare-metal) Dedicated Cluster: This consists of 16 nodes, each of which has a 2 dual core AMD processor (4 CPU cores) and 8 GB of memory. Each node runs CentOS 5.2 operating system. We used OpenMPI version 1.3.2 with gcc version 4.1.2.
2. Virtualized Cluster: Logical cluster provisioned on the physical cluster describe in Section 3. Each virtual machine (VM) has 4 CPU cores created from the native bare-metal image. The number of nodes deployed is 16.
3. HPC Cloud System: We provisioned a 128-core HPC private cloud system as summarized in Table 1.

We ran the MPI matrix multiplication application on these three configurations and measured the performance, system utilization, runtimes and overheads. Some of the results of our analysis are shown in Figures 3 through 6.
As observed from the graphs, the native cluster has the best performance and overhead values compared to the VM and HPC cloud configurations. The HPC cloud performs better than the VM in both cases.

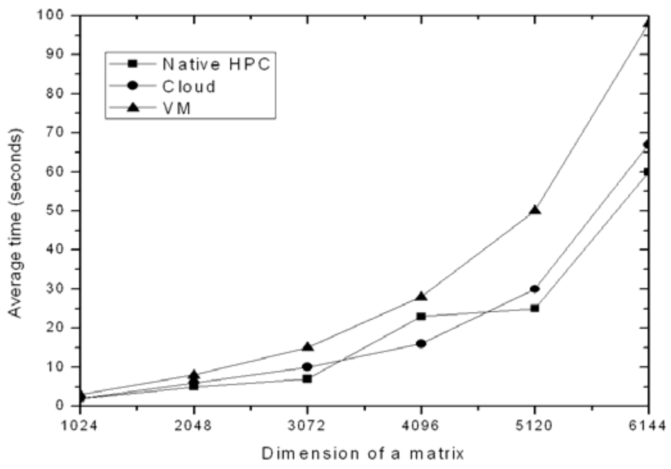


Figure 3: Performance of the matrix multiplication application using 64 MPI processes

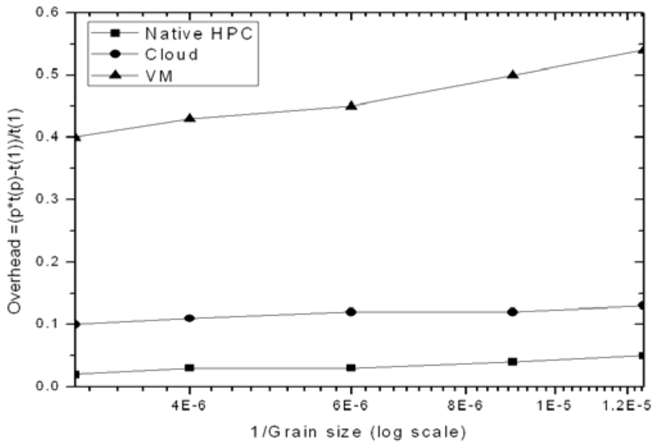


Figure 4: Total overhead of the matrix multiplication application using 64 MPI processes

It is worth noting that in application where the amount of data transfer between the MPI processes is extremely low compared to the amount of data processed by each MPI process, the effect of virtualized resources on performance degradation is clearer. The HPC cloud out-performs both native and VM configurations in system utilization. The main reason for this effect was the availability of more jobs for the scheduling mechanism and the increased size of system. The cloud unifies all compute resources together to maximize compute capacity. The richer and higher availability of application jobs provide the scheduler with more choices for maximizing the system utilization and for matching the jobs to the available compute resources.

In Figure 6, we observe that the application runtimes in both the native HPC and cloud configurations are similar and lower than that of the VM configuration.

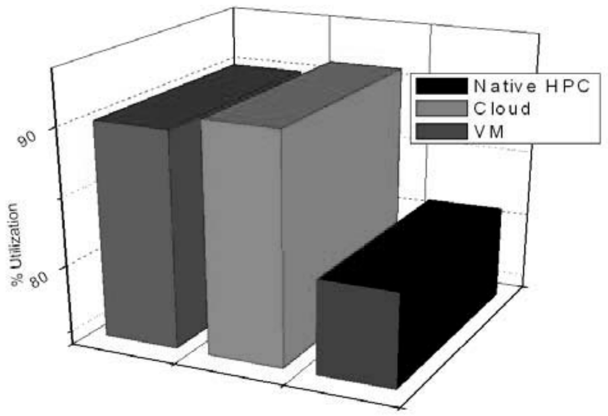


Figure 5: System utilization chart to compare between the different environments for the matrix multiplication application

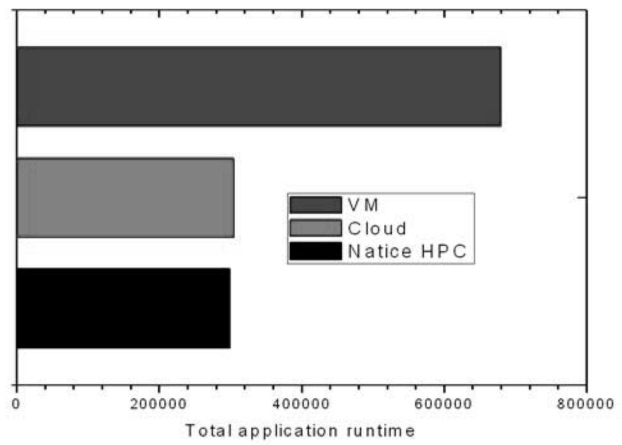


Figure 6: Total application runtime for the different infrastructure environments

6 Conclusion and future work

Cloud computing infrastructure and technologies provide immense opportunities for HPC and scientific computing. A private cloud, (i.e. a cloud within an organization) provides the most viable option of moving HPC to the cloud. Cloud technologies work well for most scientific computations. Their support for handling large data sets, the data-centric approach to computation, and the better Quality of Service provided make it extremely attractive compared to the traditional systems.

Virtualized resources, (i.e. as provisioned by public cloud vendors) pose a problem and can be avoided for now. However, cloud structures can benefit from using VMs for easier system provisioning. Virtualization solutions aimed at enabling native application performance has to support high speed networking such as Infiniband.

Future work will include adding virtualization to the clouds. The goal will be to provide native interconnect performance from the VMs. Extending this to the public cloud will also add challenges which include checkpoint methodologies.

7 References

[1] Amazon Elastic Compute Cloud (EC2), http://aws.amazon.com/ecs2/

[2] Amazon Simple Storage Services (S3), http://aws.amazon.com/s3/

[3] GoGrid Cloud Hosting, http://www.gogrid.com/

[4] Microsoft Azure, http://www.microsoft.com/windowsazure/

[5] Google App Engine, http://code.google.com/appengine/

[6] J. Ekanayake and S. Pallickara, “MapReduce for Data Intensive Scientific Analysis,” Fourth IEEE International Conference on eScience, 2008, pp.277-284.

[7] Apache Hadoop, http://hadoop.apache.org/core/

[8] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, “Dyrad: Distributed data-parallel programs from sequential building blocks,” European Conference on Computer Systems, march 2007.

[9] K. Keahey, I. Foster, T. Freeman, and X. Zhang, “Virtual Workspaces: Achieving Quality of Service and Quality of Life in the Grid,” Scientific Programming Journal, vol 13, No. 4, 2005, Special Issue: Dynamic Grids and Worldwide Computing, pp. 265-276.

[10] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, “The Eucalyptus Open-source Cloud-computing System,” CCGrid'09: the 9th IEEE International Symposium on Cluster Computing and the Grid, Shanghai, China, 2009.

[11] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A.Warfield, “Xen and the art of virtualization,” In Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles (Bolton Landing, NY, USA, October 19 - 22, 2003). SOSP '03. ACM, New York, NY, 164-177. DOI= http://doi.acm.org/10.1145/945445.945462

[12] M. J. Chin, S. Harvey, S. Jha, and P. V. Coveney, “Scientific Grid Computing: The First Generation,” Computing in Science and Engineering, vol. 7, 2005, pp. 24–32.

[13] C. Evangelinos, C. N. Hill, “Cloud Computing for Parallel Scientific HPC Applications: Feasibility of Running

Coupled Atmosphere-Ocean Climate Models on Amazon's EC2,” Cloud Computing and Its Applications 2008 (CCA-08), Chicago, IL. October 2008.

[14] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good, “The cost of doing science on the cloud: the montage example,” Proc. Of the 2008 ACM/IEEE conference on Supercomputing (SC’08). Piscataway, NJ, USA: IEEE Press, 2008, pp. 1–12.

[15] R. Buyya, C.S. Yeo, and S. Venugopal, Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities, Keynote Paper, in Proc. 10th IEEE International Conference on High Performance Computing and Communications (HPCC 2008), IEEE CS Press, Sept. 25–27, 2008, Dalian, China.

e-Biochem: A Web Based Bioinformatics & Life Science Job Management System for a HPC Cluster

Jayant Mukherjee[1], **Prabha Garg**[2], **and Avadhesh Mittal**[3]
[1]Project Manager, Altair Engineering, Bangalore, Karnataka, India
[2]PharmaGrid Lab, National Institute of Pharmaceutical Education & Research, Mohali, Punjab, India
[3]Sales Manager, Altair Engineering, Bangalore, Karnataka, India

Abstract - *Research Labs & Organizations engaging in a setup of complex HPC cluster needs to track the usage of HPC by end users. The complexities & difficulties involved in this setup for scientists & students are to know the commands associated to the application as well as HPC environment. This increases the reluctance of using the grid environment and user prefers to use standalone applications thus making the HPC investment highly costly.*

*This paper provides the feature of e-Biochem, a based Bioinformatics & Life Science job management portal integrated with PBS Professional. Users use the portal to submit their job by selecting applications, cpus, O/S & priority with pre-defined rules thereby making the HPC setup more user friendly and cost benefit. It also provides a platform for universities and labs to provide a **BioFarmsAsCloud(BFAC)** environment for other organizations/labs to use the setup continuously and optimally*

Keywords: e-Biochem, PBS, BFAC, MPI, HPC

1 Introduction

Bioinformatics and Life Science field involves usage of many computational types of software which are used for various purposes. In a typical laboratory of these departments, one can find a HPC setup which could be a grid farm or SMP boxes.

Users are generally allowed to access the application via remote connection and use the standard in-built Graphical User Interface (GUI) and/or command-line argument of the application for submission of the input file/s. The jobs are then get generated & submitted to the HPC/SMP boxes.

However, the drawback in this process is that user should remember the complex commands of the applications, the First-In-First-Out (FIFO) jobs and hence making all jobs in thread. This way it reduces the chances of putting high priority or high CPU jobs unaltered, unless a resubmission with new criteria is selected.

If the grid farm uses a grid computing software, the knowledge of understanding the right arguments and remembering the complex arguments is another vernacular problem.

In order to handle the above cases, a web based Web 2.0 compliance based solution is developed, called e-Biochem, which acts as interactive mode of job submission and management of jobs for the end users thereby avoiding the complexities involved in the back-end high performance computing.

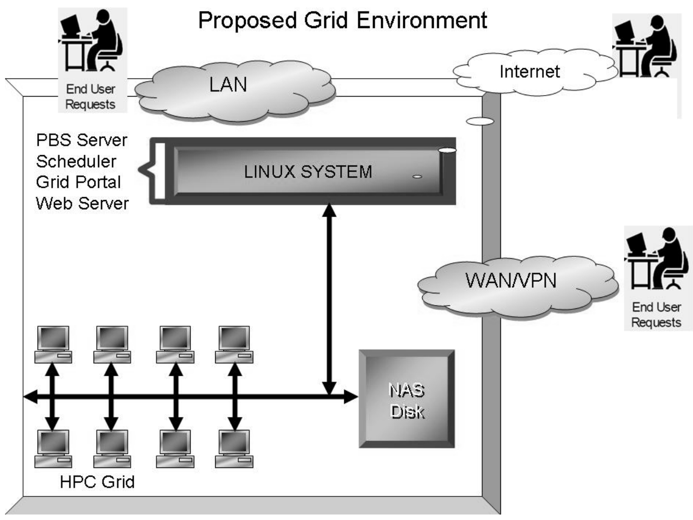


Figure 1: e-Biochem & HPC setup at NIPER

2 About e-Biochem

Currently e-Biochem is a complete web based solution using Apache Tomcat as application server, database as Derby & grid infrastructure software called PBS-Professional (Portable Batch System). PBS-Professional is an Altair Product which is in itself handles all kind of complexities involved in grid computing like queuing, managing grid farms, job priority, fail-safe mode etc. The detail about the product can be found out from the official site: www.altair.com

e-Biochem communicates with PBS-Professional by creating internal batch commands which PBS can understand and communicates with PBS to give the desired output to the end user. This way, user need not to worry about the communication between what happens in a complex hpc setup and what is the end goal.

2.1 e-Biochem Architecture Details

e-Biochem is developed using Struts 2.0 Framework with following components in it:

1. Web Pages on JSP

2. Application/Web Server: Apache-Tomcat 5.0

3. Database: Derby

4. Grid Computing Software: PBS Professional 8.0

Refer Appendix I for the Flow Diagram of e-Biochem representing the BFAC model.

2.2 e-Biochem Infrastructure Features

e-Biochem supports following infrastructure features:

2.2.1 Linux Cluster :

Some applications operate only on a linux environment. e-Biochem works on Red Hat Linux operating system.

2.2.2 Windows Cluster :

Some applications operate only on a Windows environment, varying from Win2003, XP and so on. Also many biochemical applications need GUI mode on Windows machine to create input deck for the job. e-Biochem provides the feature of remote connection through web portal for creation of the deck and submitting it seamlessly to the cluster.

2.2.3 Heterogeneous Cluster :

Applications like OpenEYE needs both Linux & Windows based support. e-Biochem helps in getting rid of complexities of finding which grid is available for which OS for submitting job. e-Biochem achieves this feature by using the Peer-to-Peer mechanism provided by PBS.

2.2.4 Single-Sign-ON (SSO) {LDAP/ADS Integration} :

In many organizations, departments have their own LDAP dedicated to a domain. It is preferable for the end user to have integration with LDAP and the portal in order to keep the session on as soon as the user logs in to a workstation. e-Biochem uses the session management concept and helps in seamless integration with the existing LDAP system of the department.

2.2.5 Optimal Utilisation of Bio-Farms :

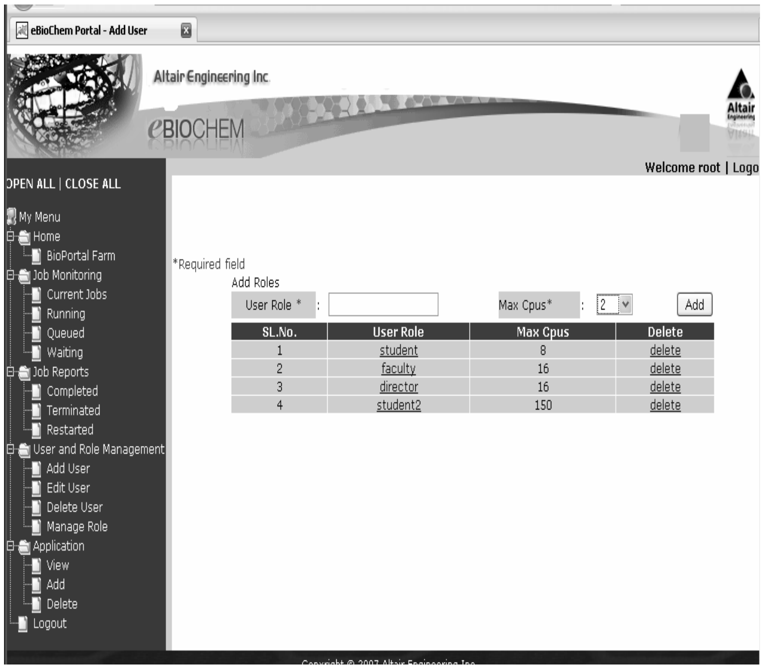
e-Biochem gives administration group to access which nodes are alive and which nodes are in sleep mode in the bio-farms. In case a node giving problem during job submission, an administrator can view the report to know which node/s had failed during job submission on a timely basis and can take necessary action to rectify the cause.

2.3 e-Biochem Portal Features

2.3.1 Access Control as per Groups/Roles:

- Administrator is given a dedicated admin page for creation of users and groups and assigning the users to the respective groups.

- In this case, three groups: Students, Faculty, Director has been created and users are added to the group.



*Required field

Add Roles

User Role * :

Max Cpus* : 2

SL.No.	User Role	Max Cpus	Delete
1	student	8	delete
2	faculty	16	delete
3	director	16	delete
4	student2	150	delete

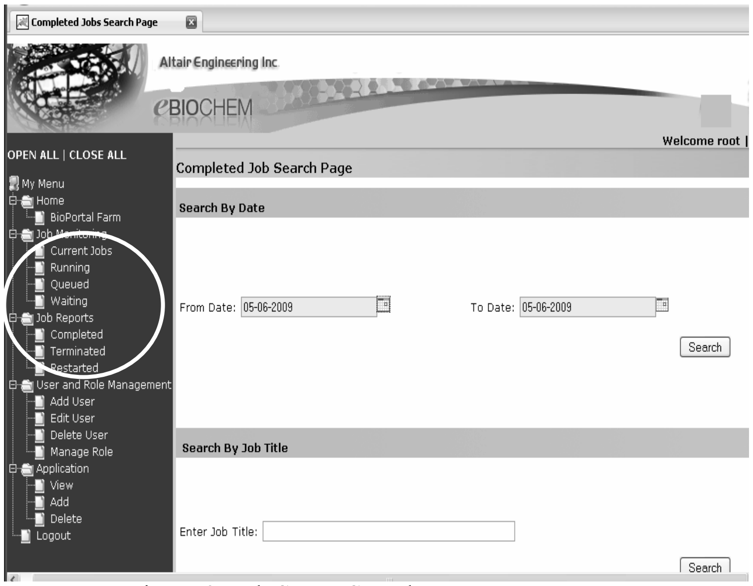
Figure 2: Users/Groups Registration on e-Biochem

2.3.2 CPU Dedication Per Role:

Many laboratories prefer to use the CPUs in optimal manner and it is always recommended that a rule can be imposed on the groups on what level of CPUs the users belonging to that group can request. This way, the CPUs available in the grid can be used in better way and bio farms are available for different kind of job submission (Figure 3)

2.3.3 Job Status:

A job goes through multiple states:: **Submitted** → **Queued** → **Started** → **Terminated** → **Paused** → **Restarted** → **Finished**. It is always useful for end user and admin to know a job status. e-Biochem takes into consideration the job ID which is a unique identification number for any job and search the response from PBS about its status in conjunction with the database about its different parameters to give the status of the job to the end user. (Figure 4)



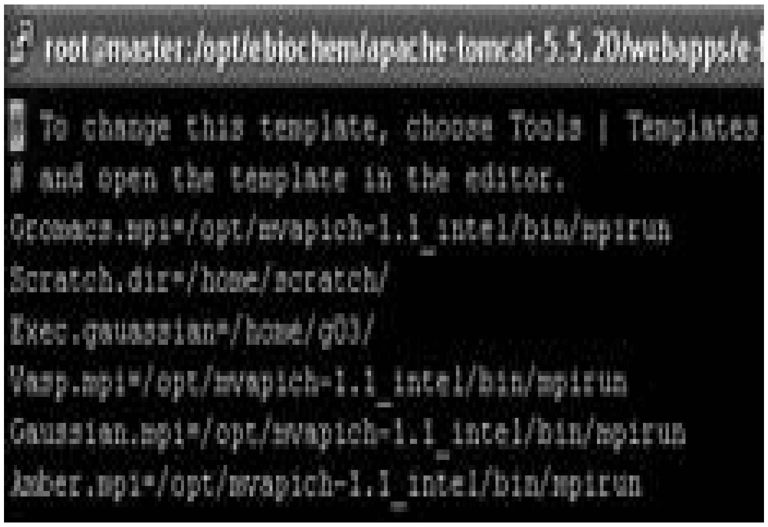
From Date: 05-06-2009 To Date: 05-06-2009

Figure 3: Job Status Search Page

2.3.4 Configuration of MPI Paths:

Jobs in a cluster environment can be classified as serial or Parallel job. A Serial job follows a FIFO concept, except when a queue-priority rule is imposed whereas Parallel job means a job is broken down into multiple job arrays or parallel jobs which run on different nodes and gives the final merged output. MPI (message parsing interface) libraries of an application provide the feature of breaking down the requested job into multiple parallel jobs and reunite the same after job completion.

e-Biochem integrates with the MPI libraries and the prerequisite is that the respective MPI libraries of the application must be registered by an administrator in order to make PBS understand the same. The best example is of Linda MPI integration with e-Biochem which is a MPI library for Gaussian application (refer Figure 5).



```
root@master:/opt/ebiochem/apache-tomcat-5.5.20/webapps/e-
# To change this template, choose Tools | Templates
# and open the template in the editor.
Gromacs.mpi=/opt/mvapich-1.1_intel/bin/mpirun
Scratch.dir=/home/scratch/
Exec.gaussian=/home/g03/
Vasp.mpi=/opt/mvapich-1.1_intel/bin/mpirun
Gaussian.mpi=/opt/mvapich-1.1_intel/bin/mpirun
Amber.mpi=/opt/mvapich-1.1_intel/bin/mpirun
```

Figure 4: MPI Path Registration

2.3.5 Upload/Download Multiple Files:

Generally, without a HPC setup, users remain unconnected with the input and associated files for a job. Even with a HPC setup where there is no GUI for job submission, users always need to have access to remote file server to get output files or share output files.

e-Biochem coordinate with the NFS of the HPC setup to give access to the specific output files associated to the job. These files could vary from a deck of associated files to just images.

2.3.6 View Log/Output Files from Portal:

In most circumstances, the job gives output of log file associated to the input file and application logs, a cluster software log and user/system specific logs. These logs could have size varying from KBs to GBs. Locating these files in the NFS & downloading on the local machine is a painstaking task.

Using Java APIs of TCP/IP file transfer on need basis, e-Biochem helps in avoiding the complexities involving in search and opening the log files.

2.3.7 Restart/Pause Jobs

When a job is in the queue status, it can always be altered by first stopping the request given to the HPC software. However, if the job starts running and user knows that it will go into an infinite loop or will end up giving erroneous result, users need to be able to pause the job to see the possible errors and restart by correcting the parameters. However this is an application feature and not of the HPC.

e-Biochem relies on this mechanism and if an application supports restart/pause, e-Biochem calls the respective commands through PBS to carry out the same action.

2.3.8 Support for more than one executable for application:

In a workflow kind of execution, where input of one file goes as output to other (like CPMD 3 stage application), many executables need to be called in sequential manner and checkpointing must be done before knowing if the job is successfully executed or not.

e-Biochem can call different executables for the same job if it involves a workflow kind of job submission and finally gives the output to the end user for each executable.

2.3.9 Automatic Scratch File Deletion

Many jobs result in redundant, unwanted and intermediate scratch files which occupy larger space in the NFS. The IT administrator needs to clean up such files and it is very difficult for them to find out which files need to be retained and which to be discarded when the output and scratch files reside in the same location on the NFS.

Sometimes the non-deletion of these files result in future failures of the jobs as the user might find the unavailability of the file space in the infrastructure.

e-Biochem has its own epilogue script which is registered on PBS. On request of the deletion of scratch files, these epilogue scripts target the unwanted files and free the space for the next job.

2.3.10 GUI Mode intégration through Active Console Window on Server Side:

It is to be noted that applications are not always batch supported. Some applications need input deck creation and then submission to the solvers. However in a complex infrastructure setup with limited licenses of the application, end users prefer to do remote connection to the server, trigger the input application software and utilize on need basis.

The disadvantage is that if in case multi users target the same machine, remote connectivity could be an issue and also role and user based requested priority can not be avoided with the proposed situation.

e-Biochem uses xHost concept of calling an active desktop of the remote server from the web page which gives the user same look and feel. The user can then create the input deck, make it ready for submission on the bio farm and open the portal, browse the recently created input file and submit it to the grid

3 Applications Integrated with e-Biochem

Following table shows the list of applications currently integrated with e-Biochem. These applications are available in Pharma Grid lab of NIPER and have been tested for all the cases as per the features mentioned in column number 2.

Table 1: List of Applications integrated with e-Biochem with Feature

Sl. No	Application Name	Job Type	Cluster Features
1	Amber	Parallel with MPI	Eplilogue Script for cleanup of Unterminated MPI Processes
		Serial	File Upload multiple Files using Zip
			File Download Multiple Files using Zip
			File View
			Parameters Selection for Job Submission from Web Browser
2	Gaussian	Parallel	Linda Integration
		Distributed	Automatic Restart of Job
			Automatic creation and clearing of Scratch Directory
			Input File Modification before Restarting
			Eplilogue Script for cleanup of Unterminated MPI Processes
			File Upload multiple Files using Zip
			File Download Multiple Files using Zip
			File View
			Parameters Selection for Job Submission from Web Browser
3	Gromacs	Parallel	Integrated with MD Run
			Eplilogue Script for cleanup of Unterminated MPI Processes
			File Upload multiple Files using Zip
			File Download Multiple Files using Zip
			File View
			Parameters Selection for Job Submission from Web Browser
4	MATLAB	Parallel	Eplilogue Script for cleanup of Unterminated MPI Processes
		Serial	File Upload multiple Files using Zip
			File Download Multiple Files using Zip
			File View
			Parameters Selection for Job Submission from Web Browser

Sl. No	Application Name	Job Type	Cluster Features
5	NAMD	Parallel	Eplilogue Script for cleanup of Unterminated MPI Processes
			File Upload multiple Files using Zip
			File Download Multiple Files using Zip
			File View
			Parameters Selection for Job Submission from Web Browser
6	VASP	Parallel	Eplilogue Script for cleanup of Unterminated MPI Processes
			File Upload multiple Files using Zip
			File Download Multiple Files using Zip
			File View
			Parameters Selection for Job Submission from Web Browser
7	CPMD 1st Stage	Parallel	Eplilogue Script for cleanup of Unterminated MPI Processes
			File Upload multiple Files using Zip
			File Download Multiple Files using Zip
			File View
			Parameters Selection for Job Submission from Web Browser
8	CPMD 2nd Stage	Parallel	Eplilogue Script for cleanup of Unterminated MPI Processes
			File Upload multiple Files using Zip
			File Download Multiple Files using Zip
			File View
			Parameters Selection for Job Submission from Web Browser
9	CPMD 3rd Stage	Parallel	Pbs Script and GUI integration
			File View
			Parameters Selection for Job Submission from Web Browser
10	TurboMol	Parallel	Initial Input Screen is Ready
			PBS Script is integrated
			Parameters Selection for Job Submission from Web Browser

4 Conclusions & Further Work

In this paper, we presented a Web2.0 framework where Life and Chemical Science and Bioinformatics applications which are installed and used for job submission on a HPC cluster with PBS can now be used effectively. The product installed on NIPER has given the scientists & research students to now effectively use the Job Portal system without engaging into complex grid and application commands. The portal helps to effectively use the cpus, job farms and make the output faster by realizing the job failures and restart through web.

Portal feature gives flexibility of job submission from any mode having secured internet access. It also provides an environment for Cloud Computing called BioFarmsAsCloud(BFAC) for other labs to use the feature.

Going forward, the other applications like OpenEYE, Schrodinger & other well known solvers used in the scientific world will be integrated. .

5 References

[1] Sites & Support documents of all applications for batch commands

[2] Sites & Support documents of PBS GridWorks by Altair

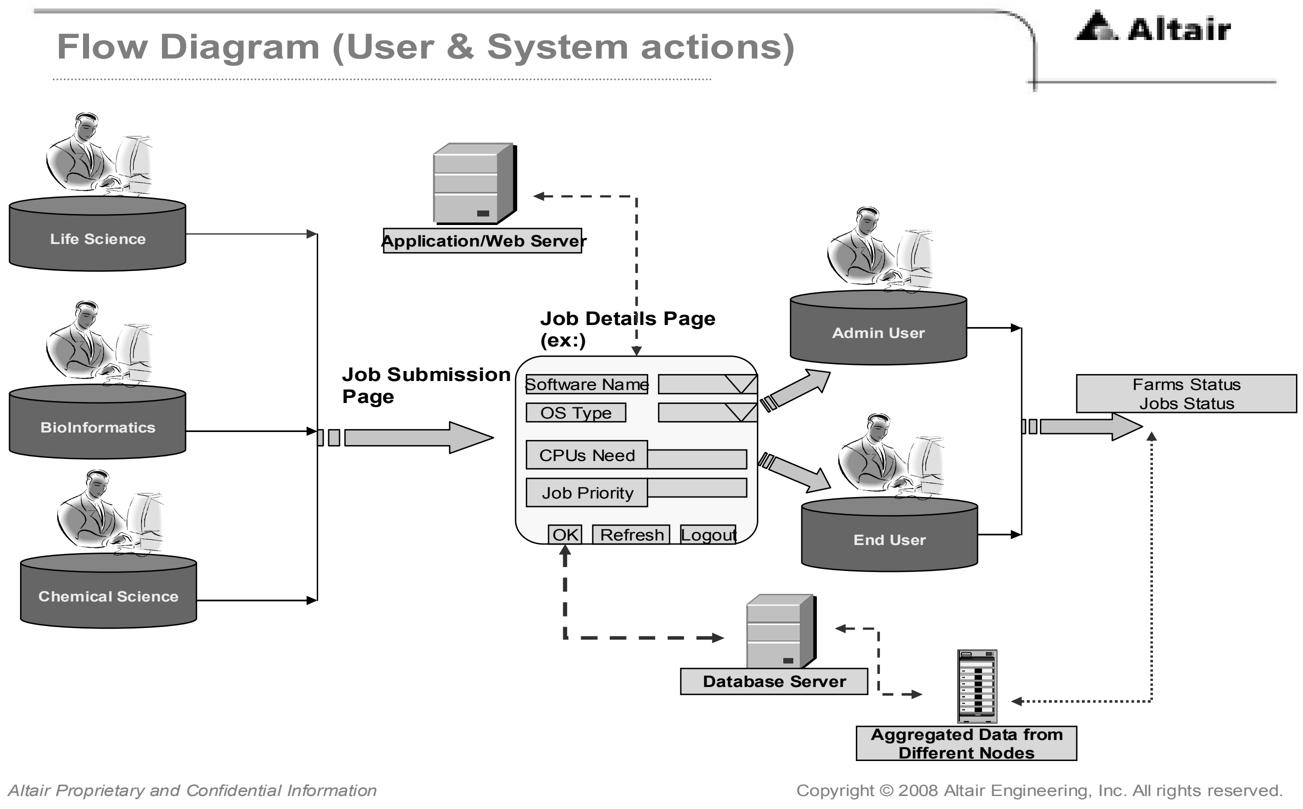
6 Appendix

6.1 Appendix 1: User & System Interaction Diagram

User from different departments located in multi-site (or different external paid labs/organizations) as per their role can start the URL and select the software for which they want to submit the job. The input deck can be browsed from local system or from a mapped network drive.

Portal gives the feature of selection of the o/s environment like if its Windows or Linux or heterogeneous environment. It also provides the action of putting high priority job with dedicated CPU for quicker and optimized utilization of bio farms. Below figure shows the typical BFAC model:

Figure 5: Flow Diagram of e-Biochem



A Latency Optimizing Load Distribution Scheme for Grid Networks

O. A. Rahmeh[1], P. Johnson[2], and D. K. Saini[3]
[1,3]Faculty of Computing and IT, Sohar University, Sohar, Sultanate of Oman
[2]School of Engineering, Liverpool JMU, Liverpool, L33AF, UK

Abstract - *Grid networks aims to utilize a high-performance computing environment to solve large-scale computational problems. To achieve such network system, the workload needs to be efficiently distributed among the computing resources accessible on Grid network. Therefore, in this paper, a distributed load balancing scheme for Grid networks is proposed. Moreover, such computing environment makes quality of service (QoS) highly desirable, though it is very complex to achieve due to the large scale of interconnected networks. The provision of network latency in a Grid environment is the topic of interest of this paper, and what we want to do is to develop a latency optimizing load distribution scheme for Grid networks. We demonstrate that introducing a latency reduction factor in the random sampling can reduce the effects of communication latency in the Grid network environment.*

Keywords: *Load balancing, Latency, Grid networks, QoS, optimization.*

1. Introduction

For decades, several techniques have been developed to utilise the wide-area distributed computers for solving large-scale problems Grid Network is an example of such network system which uses the network resources of many computers to solve large-scale computational problems [1]. Therefore, a proper load distribution and resource discovery protocols are needed to distribute the workload among the available nodes in the network to improve the overall system performance.

One of the essential features of the Grid networks is that the resources accessible in the network are distributed geographically. However, one of the fundamental challenges to run Grid applications across geographically distributed computational resources is overcoming the effects of the latency between them. While high performance clusters and supercomputers can deliver data to applications with latencies of few microseconds, latency across the wide area networks is measured typically in milliseconds. Therefore, reducing the effects of communication latency is critical for achieving good performance with Grid applications that involve significant amounts of communication. In this paper, an efficient biased random sampling (BRS) algorithm is shown to reduce communication latency in Grid networks and thus enabling the network to achieve load balancing which is scalable and reliable.

There are large amounts of research deal with different load balancing methods, and several algorithms have been proposed to address this issue [2-13]. Though, many of these methods rely on centralized methods, which can be effective in small scale networks or on specific properties of load distribution in larger networks. Also, centralized techniques are not scalable as they need high computing power and large bandwidth, network systems that depend on such techniques are un-scalable [14-15]. In addition, centralized methods are not reliability since they have a single point of failure.

2. Proposed load distribution mechanism

For efficient usage of resources in Grid networks, one would want to distribute processes as evenly as possible, so that no node is more loaded than the others. Therefore, generating a dynamic network system that provides a balanced load distribution and effective resource discovery is required.

To generate this dynamic system, the in-degree distribution of nodes in the network must be investigated. A node's in-degree refers to the free resources of the node. The job assignment and resource updating processes required for load balancing are encoded in the network structure. Therefore, when a node receives a new job, it will remove one of its edges to decrease its in-degree. Likewise, when the node finishes a job, it will add an edge to itself to increase its in-degree.

In [16-18], we proposed a distributed load balancing framework for large scale networks. The addition and deletion of node's edges is performed via Fitted Random Sampling (FRS). FRS is the process whereby the nodes are randomly chosen with equal probability. The sampling starts at some node, and at each step, it moves to a neighbour of the current node, which is chosen randomly according to an arbitrary distribution.

Analogous schemes have been used for load balancing which yielded some considerable results [11, 18]. However, the proposed load distribution scheme has an improvement as the generated network structure is dynamically changed to effectively distribute the workload. Moreover, no monitoring is needed since it is encoded in the network structure.
In addition, the number of sampling steps will be constrained to a fixed length, and the nodes will be chosen according to a predefined criterion.

In this paper, a *biased random sampling* (BRS) load distribution technique is proposed which improve our previous technique by introducing a latency reduction factor in the random sampling to reduce the effects of communication latency in Grid networks.

The BRS scheme can be easily implemented in Grid networks. The desired network system can be built directly on top of the physical transport layer and use the Grid Network as its underlying network. Consequently, the network doesn't need to consist of physical links between nodes; the edges can be a routing table that provides the actual physical connections or the feasible routes between the nodes in the underlying physical layer. Additionally, the network can be implemented by using fast transport protocols sockets (such as UDP sockets) to represent the edges of the network with minimum overhead. Each node will have local information about its status which can be used for load distribution and resource allocation.

For simulations, a network system with *N* nodes is created. Each node in the network is a computer with power equal to its maximum degree. One unit of power can process a unit of load in each unit of time. Two types of simulation experiments were carried out. The first experiment considered the CPU power alone as the key factor for load balancing. In the second experiment, the geographical distance (communication delay) is added as a second factor for load balancing. Nodes' edges will be inserted or deleted to keep the nodes' in-degree proportional to its free resources.

Details of the above edge's addition and deletion process above can be found in [16, 17]. This edge's addition and deletion process will model the change in the network's load, and the amount of free resources available for the nodes will present the network load distribution status. Simulation results will be used to verify the scalability and reliability of the proposed load distribution scheme.

3. Evaluation and simulation results

Simulation results have been used to analyse and discuss the performance of the load-balancing algorithm and to determine the length of random sampling required to achieve the required load balancing. Then, we evaluate the scalability and reliability of the algorithm under several conditions. We also study the effect of modifying the random sampling by including localization information on the average communication latency of the network.

3.1. The Load balancing performance

Here, we discuss the performance of proposed load balancing mechanism under ideal conditions, where all nodes have the same resources. The simulation results prove that the proposed mechanism creates regular networks.

Figure 1 shows the simulation results for the in-degree distribution of the network plotted as the network evolved through different time slots (*T*), which shows the process of reaching the load balancing. Here the time dynamics of the in-degree distributions of the network can be clearly seen. In Figure 1.a, the network is initialised in a completely random state with variance approx. *46.3*. Then, the network starts reshaping itself by balancing the load distribution among the nodes, and in-degree variance continues to decrease. Over time, the network settles down to a nearly regular graph with variance approx. *0.32* as seen in Figure 1.b. Thus, when all the nodes have the same capabilities, the network will be almost a regular graph.

Figure 2 shows the state of the in-degree variance of our network system with time as the network evolves. The network is initialised randomly; for example, it starts with an in-degree variance of approximately *42.6*. Then, the network starts reshaping with time by adding and deleting nodes' edges to reach an in-degree variance value of around *63.3*. Consequently, the network starts to settle down and the variance rapidly decreases until the network becomes almost regular with in-degree variance close to *0.38*.

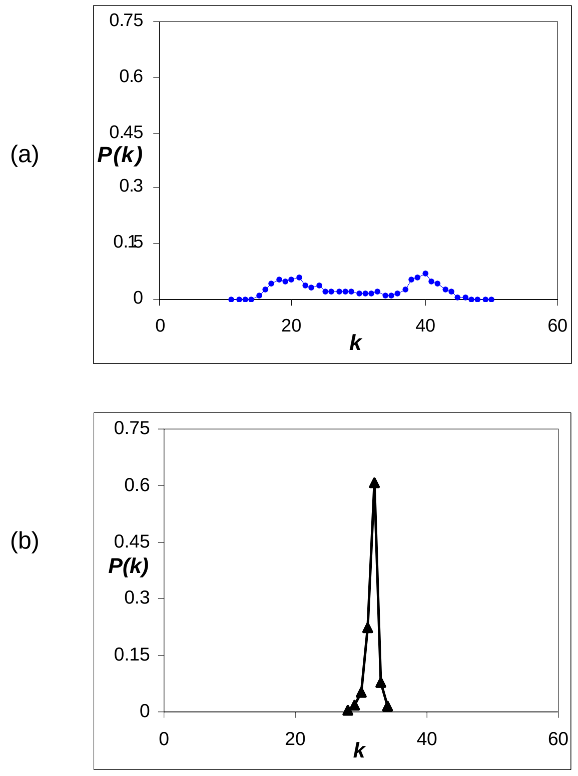


Figure 1. The in-degree distribution plotted as the network evolves over different time slots (*T*).

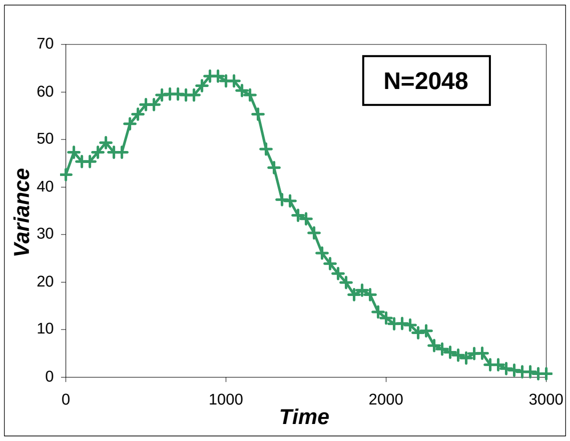


Figure 2. The variance of the nodes' in-degree distribution over Time.

3.2. Latency reduction load distribution

In reality, load balancing is not restricted to the use of resources or computing power, but also is influenced by the geographical distance between the nodes. Therefore, we included locality information into the random sampling scheme. Thus, the random walk may prefer a geographically closer node even if it is not the highest degree on the walk. We implemented this by adding the geographical distance and communication delay factors in sampling the nodes to distribute the load balancing.

Simulation results show that adding the locality factor reduced the overall network latency. We performed two experiments and recorded the average round trip latencies for each executed job.

As we can see from Figure 3, the latencies observed in the offered load using geographic-aware scheme are reduced by at least *22%* on average from that observed for the non-geographic aware scheme. For example, for networks with *512* nodes distributed with a radius of *1000km*, the overall average latency decreased from *92.58ms* to *70.17ms*. Moreover, we observed that latencies for individual loads by using this algorithm will always remain close to the average latency with no big overshoots (fluctuations), which make the network stable and reliable and a suitable environment for applications that require specific quality of service.

Simulations have been carried out to investigate the number of steps required to sample the network to reach the required load distribution. It has been observed that the performance of the load distribution algorithm improves with increasing the sampling length. It has been observed that when the random sampling is very short, the load distribution is not efficient with high variance. Moreover, when the sample length is too large, the reduction in the in-degree variance is too small. Nevertheless, when the random sampling length is around *log(N)* on networks with *N* nodes, the in-degree variance is near to the variance for balanced networks [18].

To examine the efficiency of adding locality factor on load balancing, simulation results were analysed for the network under consideration over several sampling lengths, and compared with the original scheme.

As we can see from Figure 4, the Geographic-aware load balancing requires few additional sampling steps to achieve the required variance for balanced network. For example, for a network with *2048* nodes, a random sampling length of *16* was sufficient for the original scheme, while the Geographic-aware scheme required a random sampling length of *20* to balance the load distribution, which still in order of *log(N)*. However, this increase in number of steps is negligible compared to the size of the network.

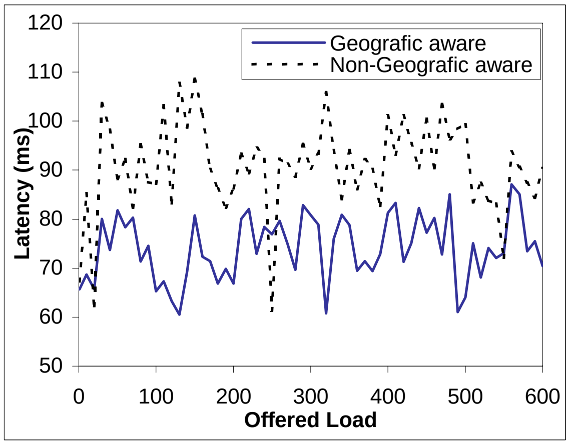


Figure 3. The average round trip latencies observed for finished jobs in a network with *N=512*.

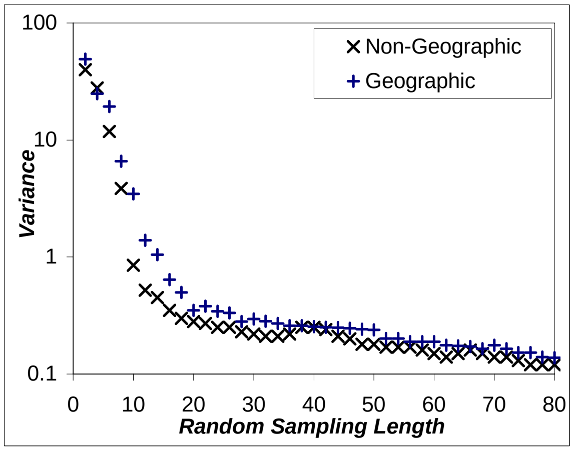


Figure 4. Comparison of the variance vs. random sampling length for a network with *N=2048*.

3.3. Performance Comparison

To evaluate the performance of the proposed biased random sampling algorithm, we examined two important performance measurements in distributed systems: the total job throughput achieved and the bandwidth required by the load distribution mechanism. Then we compared the performance the biased random sampling scheme with the performance of the centralised mechanism.

Here, we analysed the bandwidth consumed by the biased random sampling (BRS) algorithm and we compared it with the centralised algorithm. As we can see from Figure 5, the central server algorithm requires less total bandwidth than the biased random sampling algorithm. In the centralised scheme, the central node has to know the load status in each of the nodes that in the network. Therefore, the central node needs to periodically check the status of every node in the network, and the nodes have to inform the central node if they finished executing the jobs so that the central node can update network load status. As a result, the total bandwidth consumed by the network is in order of O(*N*).

For the biased random sampling algorithm, each node that initiates a new job must initiate a random sampling to search for a node to give it the job. And since the random walk will be O(*logN*) length, the total bandwidth of the walk will be in order of O(*logN*). Therefore, for *N* nodes network, the total bandwidth consumed by the biased random sampling algorithm will be in order of O(*NlogN*), which is greater than the total bandwidth consumed in the centralised scheme.

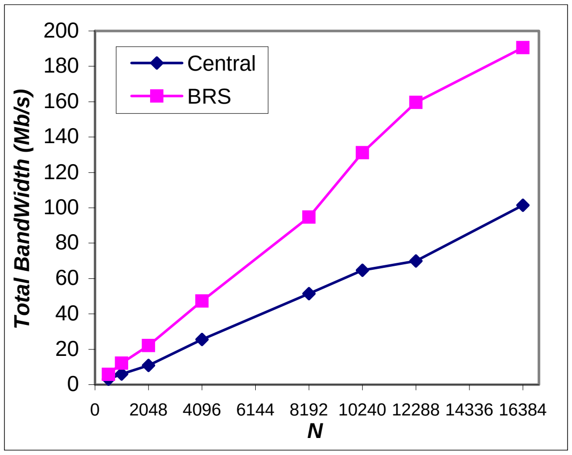


Figure 5. Simulation results for the total bandwidth consumed in different network sizes.

However, the biased random sampling scheme decreases the bandwidth consumed by any node in the network, as shown in Figure 6. The central node in the centralised scheme is engaged in all jobs and handshaking transfers. Therefore, the central node consumes a O(*N*) bandwidth. For the biased random sampling algorithm, the bandwidth required for each node depends on the node in-degree and on the number of jobs it initiates. Thus, if the *N* nodes in the network use the total network bandwidth uniformly, then each node in the network will consume a bandwidth in order of O(*logN*).

Although the total bandwidth consumed in the network is a significant performance measurement, the bandwidth consumed by any single node in the network can be a major bottleneck for large-scale networks.

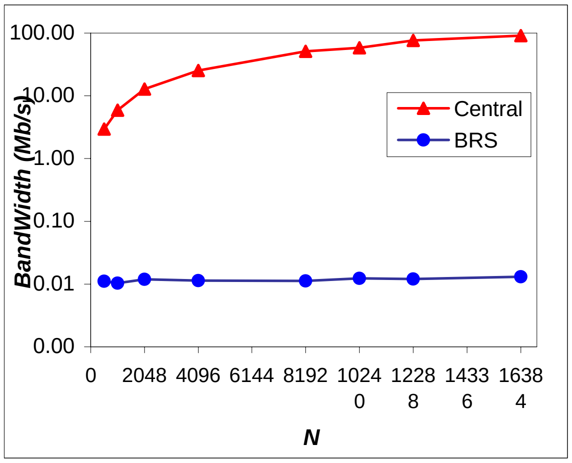


Figure 6. Simulation results for the average bandwidth consumed by single nodes for different network sizes.

Another important performance criterion in distributed networks is the system throughput. Throughput is the number of jobs completed during a specified period of time. The objective here is to have the maximum amount of completed jobs (large amount of throughput). Therefore, we analysed the total throughput achieved by the biased random sampling algorithm and compared it with the central system.

For the simulations, the nodes in a network have equal capabilities, and the job sizes and arrival rates are Poisson distributed. Moreover, we considered the effect of communication delay on the total throughput performance by distributing the nodes in a network of *1000* miles radius area with *10Mbps* communication link speed. The throughput achieved by the biased random sampling scheme is very close to the throughput achieved by the optimal centralised scheme. The total throughput for biased random sampling algorithm is only around *3%* worse than the total throughput of the central algorithm, with the advantage of being a distributed load-balancing scheme.

To further measure the efficiency of the proposed biased random sampling mechanism for load balancing in various situations, the simulations could be extended to include heterogeneous nodes and cases where jobs may require certain quality of service such as communications bounded, distance sensitive, and time bounded services. Examining how these considerations will affect the efficiency of load balancing is a topic for future work.

4. Conclusions

A scalable and reliable biased random sampling scheme for distributing the workload between the nodes on Grid networks is proposed in this paper. The generated network system is scalable, self-organised, robust, and depends only on local information for load distribution and resource discovery. The developed load-balancing scheme is based on biased random sampling to assign the jobs and to update resource's availability. Therefore, load balancing is achieved without the need to monitor the nodes for their resources availability.

Achieving a proper quality of service (QoS) in Grid networks highly desirable. However, it is not easy to be achieved due to the large scale of interconnected networks in such large scale systems. Therefore, a latency optimizing load distribution scheme for Grid networks has been developed and presented this paper.We demonstrated that introducing latency reduction factor in the random walk sampling could reduce the effects of communication latency in Grid network environments.

A number of improvements to the proposed load distribution mechanism and generalizations of our model deserve further study. We plan to extend this work to support heterogeneous systems and cases where workload may require special quality of services. This will help us in understanding how these factors would affect on the nodes' in-degree and load distribution in the network.

5. References

[1] I. Foster, & K. Kesselman, *The Grid: Blueprint for A Future Computing Infrastructure*, Morgan Kaufmann, 1999.

[2] R. Lüling , B. Monien & F. Ramme, "A Study of Dynamic Load Balancing Algorithms", *Proceedings of the Third IEEE SPDP*, 1991, 686-689.

[3] L. P. Peixoto, "Load Distribution: A Survey", Technical Report. Dept. De inf, Escola De Engenharia, Universidade Do Minho, 1996.

[4] Y. Murata, *et al.*, "A distributed & cooperative load balancing mechanism for large-scale P2P systems", *SAINT-W*, USA, 2006.

[5] M. Mitzenmacher, "The Power of Two Choices in Randomised Load Balancing", *IEEE Transactions on Parallel Distribution Systems*, 2001, 12(10).

[6] Y. Drougas, T. Repantis, and V. Kalogeraki, "Load Balancing Techniques for Distributed Stream Proceszing Applications in Overlay Environments", *ISORC'06*, USA, 2006.

[7] J. Bustos, D. Caromel, "Load Balancing: Toward the Infinite Network", 12th Workshop on Job Scheduling Strategies for Parallel Proceszing, France, 2006.

[8] M. M. Theimer, & K. A. Lantz, "Finding Idle Machines in A Workstation-Based Distributed System", *IEEE Transactions on Software Engineering*, 1989, 15(11).

[9] D. Oppenheimer, J. Albrecht, D. Patterson & A. Vahdat, "Scalable Wide-Area Resource Discovery", *Technical Report*, CA, USA, 2004.

[10] R. Subramanian, & I. Scherson, "An Analysis of Diffusive Load Balancing", *Proc. of the sixth Annual ACM Symposium on Parallel Algorithms & Architectures*, ACM Press, 1994.

[11] A. Montresor, H. Meling, & O. Babaoglu, "Messor: Load-Balancing Through a Swarm of Autonomous Agents", *First Intl. Workshop on Agents & P2P Computing*, Italy, 2002.

[12] M. Litzkow, M. Livny, & M. Mutka, "Condor: A Hunter of Idle Workstations", *Proceedings of the Eighth International Conference of Distributed Computing Systems, 1988.*

[13] B. Yagoubi, and Y. Slimani, "Task Load Balancing Strategy for Grid Computing", *Journal of Computer Science*, 3 (3), 2007, 186-194.

[14] R. Lüling, & B. Monien, "A Dynamic Distributed Load Balancing Algorithm with Provable Good Performance", *SPAA '93*, ACM Press, NY, USA, 1993.

[15] O. Kremien, & J. Kramer, "Methodical Analysis of Adaptive Load Sharing Algorithms", *IEEE Trans. On Parallel Distribution System*, 3(6), 1992.

[16] Rahmeh, O. A., Johnson, P., and Lehmann, S., "A Fitted Random Sampling Scheme for Load Distribution in Grid Networks", *the International Journal of Information Technology*, V24, ISSN: 1307-6884, 2007.

[17] Rahmeh, O. A., Johnson, P., "Towards scalable and reliable Grid Networks", *IEEE/ACS International Conference on Computer Systems and Applications*, 4 April 2008, ISBN: 978-1-4244-1967-8, Page(s):253–259, Doha, Qatar, 2008.

[18] Rahmeh, O. A., Johnson, P., and Taleb-Bendiab, A., "A Dynamic Biased Random Sampling Scheme for Scalable and Reliable Grid Networks", *the INFOCOMP Journal of Computer Science*, Vol. 8, No. 1, ISSN: 1807-4545, 2008.

[19] C. Avin, & C. Brito, "Efficient and Robust Query Proceszing in Dynamic Environments Using Random Walk Techniques", *Proc. of the third Intl. Symp on Info. Proceszing in Sensor Networks*. ACM Press, 2004.

SESSION

GRID UTILITIES, ENVIRONMENTS, TOOLS, POLICIES, SECURITY ISSUES

Chair(s)

TBA

A User-Centric Authentication for Advanced Resource Reservation in Mobile Grid Environments

Cristiano C. Rocha[1], **Matheus A. Viera**[1], **Miriam Capretz**[2], **Michael A. Bauer**[3], **Iara Augustin**[4], **and M.A.R. Dantas**[1]
[1]Department of Informatics and Statistics, Federal University of Santa Catarina, Florianópolis, SC, Brazil
[2]Department of Electrical & Computer Engineering, University of Western Ontario, London, ON, Canada
[3]Department of Computer Science, University of Western Ontario, London, ON, Canada
[4]Informatics Graduation Program, Federal University of Santa Maria, Santa Maria, RS, Brazil

Abstract - *In this paper, it is presented an authentication architecture utilizing a lightweight user-centric approach. The main goal of this research is to provide mobile users a facility to perform advanced resource reservation within grid environments. As a result, the approach requires any user a previous authentication to utilize a grid resource or service. Experimental tests indicate that the proposal was successful in terms of battery power consumption, transparency to mobile users and providing some level of security to utilize advance reservation in a grid environment.*

Keywords: Mobile grids; user-centric authentication; advanced reservation

1 Introduction

Grid computing is characterized by making a variety of distributed resources, including services, devices and applications, available to a wide range of users [1]. Various organizations, both real and virtual, can make different types of resources available under dynamically changing availability constraints and with varying policies for access and use of these resources [2]. Subsequently, the resources belonging to these organizations can be accessed and combined by different users to achieve their computational goals.

In recent years, there has been a movement towards integrating grid computing environments with mobile computing environments [3, 4, and 5]. Consequently, mobile devices within this context are considered as grid interfaces and as grid resources. Despite the fact that the computing power of mobile devices has improved significantly in recent years, the current processing power and storage capacity found in these devices are still not enough to solve complex problems. Therefore, the present study considers the use of mobile devices as interfaces to access the resources and services of a grid from anywhere, at anytime. Our approach aims to enable users to utilize mobile devices for access to advanced reservation services with the objective of submitting individual tasks and workflows. As is the case with other access to grid services, the use of mobile devices also requires a user authentication mechanism, which is a device that allows users to adopt defined or permitted roles for access to the services and resources. As studies in area of mobility suggest, a change in the approach of performing user authentication via mobile applications, namely, from a process-oriented paradigm to a user-centered one, must be accomplished [6]. Specifically, the authentication system should recognize the user rather than the equipment that the user possesses. Moreover, because mobile devices have limited power, authentication schemes that are computationally intensive or that require substantial communication are unsuitable. This change to a user-centered approach, coupled with the limited power resources of mobile devices, imposes new requirements on the security and authentication systems for supporting the use of mobile devices within grid environments.

The current work presents an architecture that provides a “lightweight” user-centric authentication mechanism for the use of mobile devices within grid environments. In particular, its purpose is to provide the user with the full range of mobile service offered by these environments. Accordingly, our approach to authentication is in the context of providing the mobile user with access to resource reservation services.

The paper is organized in five sections. The motivation for the development of this approach is presented in Section 2. The proposed architecture is introduced in Section 3, and Section 4 presents the experimental results. Finally, conclusions and future research work are shown in Section 5.

2 Motivation

The research in [3], a previous work of our group, proposes a framework for submitting and monitoring grid computing tasks through mobile devices. In that study, there is a mechanism for managing disconnections that result from a drop in battery power or from interference in the wireless network. However, this framework poses a disadvantage in the case where a user accesses different devices during the execution of an application. In this situation, each device must perform the entire authentication process, thus reducing the battery charge and the system productivity. Furthermore, this work does not examine access to an advanced resource reservation facility in the grid environment and uses a traditional authentication where a username and password are requested in each interaction.

Advanced resource reservation in the grid computing environment has been the focus of recent research (e.g., [7], [8] and [9]). By reserving resources, a client has guaranteed access to a specific resource in the grid for a designated time period. Accordingly, [7] and [8] present mechanisms for resource reservation using the concept of co-allocation [10], which can be defined as the simultaneous use of grid resources across multi-clusters. The approach described in [7] provides resource reservation for a single resource or for a set of resources by using co-allocation. In addition, the work suggests the use of a ticket for subsequent interaction with the reservation. In [8], the authors present an approach that uses the Web Service Resource Framework (WSRF) [11] to perform resource reservation, along with the introduction of a two-phase commit protocol. Their objective is to use a non-blocking mechanism that avoids disconnection problems and can facilitate the recovery of failed processes. While [7] suggests the use of co-allocation, [9] introduces the idea of virtual resources without co-allocation. The elimination of this mechanism allows the generic integration of different types of resources and reservations with the use of temporal and spatial relationships between components.

Because of the movement from a process-based paradigm to the user-centric paradigm, some studies present requirements that must be considered for user-centric security systems. Therefore, in order to fulfill these requirements, it is necessary to analyze context-related information, such as the user's location, the time in which the interactions occur and the context-based detection of anomalous transactions [12]. Several works present complex solutions for user authentication based on the environmental context ([13], [14]). Most of these studies achieve their objectives by using several sensors in these environments. However, these proposals restrict user mobility because they are only effective within the area covered by the sensor network. For instance, [13] presents an architecture that aims to authenticate users based on the context captured by various sensors and devices present in the "smart homes" environments. Additionally, [14] proposes an infrastructure that supports large-scale mobile applications and enables the execution of these applications in different types of equipment.

3 Proposed Architecture

In this section, we propose a user-centered architecture that enables advanced resource reservation in grid environments. Our approach is partially based on the research presented in [3], where the authors propose a framework for the submission, monitoring and coordination of workflows executed in grid computing environments. All of the interaction with the grid is performed through mobile devices that are used as grid interfaces. In particular, this work suggests the possibility of adapting the execution flow to guarantee the consistency of an application in case of a disconnection occurs. This execution flow will be performed in a personalized way in the case that the mobile device is disconnected from the network. Specifically, these features are performed by *Workflow Manager, Agent* and *Mobile GUI***.**

The *Workflow Manager* module is responsible to manage the requests processes from mobile devices in a transparent way to the users. In particular, it provides an automated way of submitting jobs to the grid and it collects information about the execution of these jobs without user interaction. Thus, this mechanism contributes to the reduction of battery power consumption in mobile devices. Also, the architecture provides a mechanism for fault tolerance, especially in the case where a voluntary or involuntary disconnection of mobile devices occurs. *The Agent* module is responsible for verifying when the disconnection occurs; if the device is connected, it uses a specific time interval. Additionally, this module also manages the faults by detecting the failure and adapting the application execution flow to the environment. When a disconnection occurs, *The Agent* can adapt to the situation by continuing the execution, waiting for device establish a connection or aborting. These options are defined by the user, and the actions are performed according to the existence of dependencies from the user. Hence, through this module, the consistency of the application is guaranteed.

All grid computing interactions occur through a *Mobile GUI* interface. This interface is responsible for allowing workflow submission and permitting the visualization of final and partial results of the application in an optimized way, since only the parts of the resulting files that are considered relevant for the user are loaded in the device interface. Also, the interface contains the ability for users to monitor the application execution, so that the status of each task can be traced. In addition, users can monitor the application execution in a customized manner based on the type of mobile device and its particular screen size. Finally, the *Mobile GUI* is responsible for sending the username and password to *Workflow Manager* for authenticating the user on the grid environment. Consequently, the user has to authenticate every interaction with the grid through their mobile device.

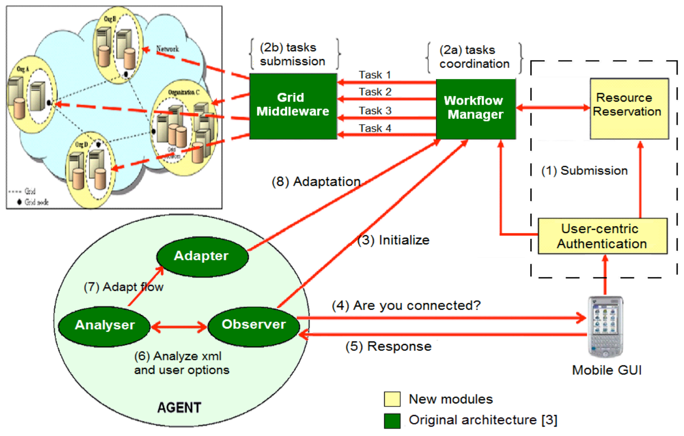


Fig 1: Proposed approach.

In addition to *Workflow Manager, Agent* and *Mobile GUI*, as proposed by Rossetto et al [3], the modules of *User-centric*

Authentication and *Resource Reservation* were added in the present approach for enabling a more secure and more efficient interaction in mobile grid environments. The new modules attempt to take advantage of the mobility offered by mobile devices while utilizing the resources from grid environments more safely and effectively. In addition to these goals, our approach attempts to improve the battery consumption of mobile devices.

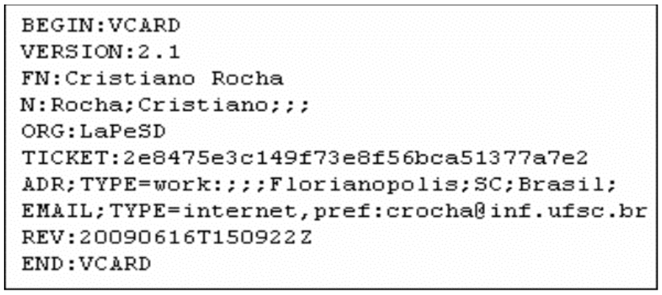
Figure 1 illustrates the research architecture of [3] with the addition of the new modules, User-centric Authentication and Resource Reservation, which are added to the proposed framework.

3.1 User-centric authentication

Through device independence, which is referred to as user-centric computing throughout this paper, the user can switch from an equipment to another without requiring a new authentication process. Thus, the potential problem of insufficient battery power can be avoided by device replacement. Moreover, the user's interaction with an application communicating with the grid environment is not interrupted, nor is it necessary to restart the authentication process. The authentication module is responsible for the interception of all service requests, which are workflow submissions or advanced resource reservations requested by an individual using a mobile device. This authentication approach obtains transparency by using a widely disseminated standard among mobile devices: the vCard standard [15], which aims at the automation of personal information sharing frequently found in a regular identification card. In this standard, the data is presented by using pre-defined meta-information that is responsible for receiving data in an organized manner and then facilitating the utilization of the data. The standard has been maintained by the Internet Mail Consortium (IMC) since 1996; the vCard standard is compatible with several existing platforms and is mainly concentrated on mobile devices, such as cell phones and PDAs.

Furthermore, the vCard standard permits the extension of meta-information in order to store other necessary data. Thus, it permits greater flexibility in the manipulation of user information and in the adequacy of each required application. The inserted information does not affect the original standard because it is ignored by the interpreter. Subsequently, the unity guarantee is maintained among various applications that involve the exchange of the same electronic identification card. Furthermore, this standard provides security for the stored information, because it offers support for digital signatures.

Therefore, the authentication system uses the vCard standard to store system-specific information. This information, such as environmental access credentials, or tickets, is represented by strings that are created by the system and that have a predetermined duration, which depends on the permissions granted to the user. For a user that has permission to reserve resources, the ticket could expire at the time of the reservation. Thus, while the lifetime of the user ticket is limited, the user can utilize other devices that have the properly formatted vCard in the system. In other words, the device has the vCard extended, so that the user does not need to reinsert his access data in the system. Figure 2 shows an example of credential represented by the vCard standard.



```
BEGIN:VCARD
VERSION:2.1
FN:Cristiano Rocha
N:Rocha;Cristiano;;;
ORG:LaPeSD
TICKET:2e8475e3c149f73e8f56bca51377a7e2
ADR;TYPE=work:;;;Florianopolis;SC;Brasil;
EMAIL;TYPE=internet,pref:crocha@inf.ufsc.br
REV:20090616T150922Z
END:VCARD
```

Fig 2: Example of an extended electronic identification card represented in the vCard format

The user-centric authentication module and its interactions with the other system modules are shown in Figure 3. The authentication process is used to prove the digital identity of the user. When the user requests a service, the system accesses the vCard in the device. (I) The *Credential Manager* selects the specific credentials of the application that are contained in the vCard and verifies if the user is on the list of active users; in other words, it verifies if the ticket is still valid. If the ticket has expired, (II) the *Credential Manager* queries the *Device Manager* to verify if the user switched to a different device. If a switch has occurred, (III) the *Location Manager* performs the functions illustrated in the activity diagram in Figure 4. Otherwise, it determines if there is an association between the user and the device used in the request. If such an association exists, the *Ticket Manager*, which controls the lifecycle of the tickets, creates a new ticket. Consequently the *Credential Manager* updates the vCard with the new ticket, and subsequently, the updated vCard is stored in the *Credential Repository*, and it is also sent to the device. Otherwise, if the user cannot be associated with the device, the system requests the login and the password of the user. (IV) If the ticket is still valid, the *Permission Manager* is queried in order to verify that the user has the permission for the requested operation. If the user has the appropriate permission, (V) the request is sent to execute the requested operation. This operation is one of the available services, along with resource reservation, reservation cancelation and the submission and monitoring of downloaded application results, the latter of which is also provided in [3]. Finally, the operation response requested by the user is returned to him/her (VI).

Moreover, since one of the main goals of the user-centric authentication module is the securityof the user's information in the environment, all messages sent between themobile devices, *Mobile GUI*, and the *Credential Manager* are encrypted. Therefore,this procedure tries to prevent malicious users from acquiring unauthorized access togrid services.

In recent years, mobile devices are able to perfom geometric models used to determine object location with geographic coordinates. These models are commonly used by location models based on the GPS (Global Positioning System). Thus, due to the advantages offered by applications that are able to facilitate location-related queries and manage accurate information regarding the location of objects, the latest generation mobile devices is being equipped with GPS to provide support for these applications. Also, it is possible to configure the accuracy of the location in order to safe battery of mobile devices. Therefore, the integration of these two popular technologies, vCard and GPS, which is still under development in the authentication system, is responsible to improve the security offered to users in the environmnent .The next section describes the functionalities of the authentication system regarding the user's location.

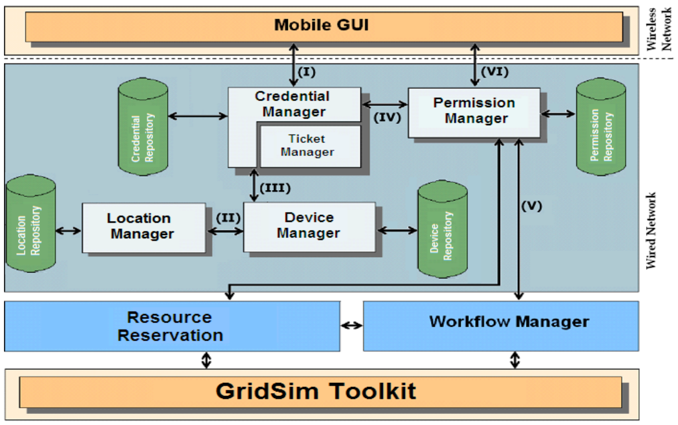


Fig. 3: Authentication module architecture and its interaction with the others modules

3.1.1 Spatio-temporal analytical model

In order to provide more reliability to the mobile grid environment as well as to minimize and detect fraudulent activities, the authentication system considers the capacity of the mobile devices to capture information regarding the spatio-temporal context of the environment where they are inserted. Thus, the *Location Manager* can classify the performed activity (event) regarding the geographic location and the time frame of occurrence of this event simultaneously. In order to formally define an event, we assume that an event is an interaction (activity) of the user with the application or environment in a certain location and at a particular time frame. Then, an event is described as:

$$E_i = < operation, time, location >$$

Therefore, the observed events in the execution of activities form a database (*Location Repository*) for the process of detecting information clusters, which translates to the behavior of users. These clusters can be classified into three broad categories: purely spatial, purely temporal or spatio-temporal. In purely spatial clusters, the occurrence is higher in some regions than it is in others, and purely temporal clusters feature the occurrence of events as being greater in a certain period than it is in others. Finally, spatio-temporal clusters occur when events are temporarily higher in certain regions than they are others. Among the models used to predict events in a spatio-temporal context, we propose the use of spatio-temporal permutation, which allows the incorporation of covariate information found in other contexts within the pervasive space. The Poisson model, which is applied to purely temporal contexts, and the Bernoulli model, which is preferably applied to spatial contexts, were both rejected because they do not consider both the location of the user and the time frame during an occurrence of an event.

According to Kulldorff [16], the spatio-temporal permutation model is based on three characteristics: *i)* detecting data clusters in space and time simultaneously; *ii)* working with only events or cases; and *iii)* applying the probabilistic model in a null hypothesis to conclude that the events follow a hypergeometric distribution.

In order to determine the regions of the clusters, it will be used the *SaTScan* tool developed by Kulldorff [16] and the statistical significance will be validated by using the hypothesis test of Monte Carlo.

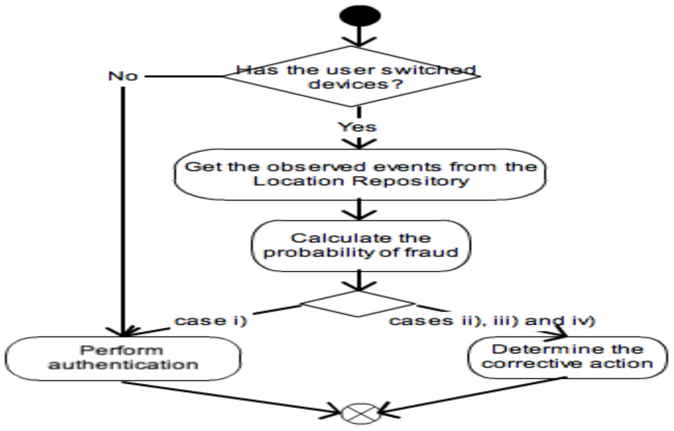

Has the user switched devices?
No
Yes
Get the observed events from the Location Repository
Calculate the probability of fraud
case i)
cases ii), iii) and iv)
Perform authentication
Determine the corrective action

Fig 4: Fraud detection activity diagram

The conditional probability of the user $P(E_a)$ allows the system to estimate what kind of activity the user was performing and what one he is currently performing when he moves from a mobile device to another one. Thus, there are four cases that can occur: *i)* the same activity in the same spatio-temporal context – it is defined as a normal execution; *ii)* same activity in different spatio-temporal contexts – it is defined as a suspicious execution, but some properties must be considered such as the velocity of mobility in order to apply the appropriate authentication policies; *iii)* different activities in the same spatio-temporal context – it is defined as a suspicious execution; and *iv)* different activities in different spatio-temporal context – it is defined as an abnormal execution. Therefore, depending on the categorization of the user, the authentication system defines which action will be taken regarding the following factors: the performed request, the malicious user, the mobile device and the potential fraud victim. The activity diagram shown in Figure 4 illustrates how the system operates when it detects device switching.

3.2 Resource Reservation

As previously discussed, the possibility of resource reservation allows mobile users to plan their future use of the grid. The resource reservation module, which is still under development, is responsible for ensuring that these reservations are maintained for future workflow submission on the grid. In addition, it enables the monitoring of reserved resources as well as recording any cancelled reservations. Figure 5 illustrates the design of the module and its functionality is described in the subsequent paragraphs.

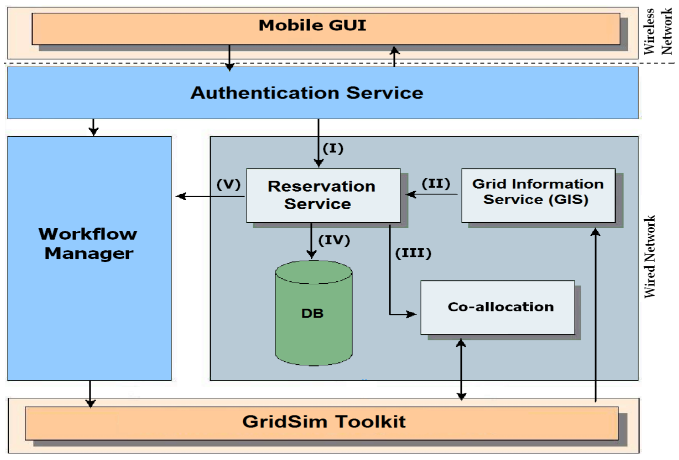


Fig. 5: Resource Reservation Architecture

First, after intercepting the request and authenticating the user, the authentication service transfers the ticket access information and the resources requiring reservation to (I) the *Reservation Service*. At the same time, the authentication module transfers the start time and end time of the reservation to the *Reservation Service*, which uses (II) the *Grid Information Service (GIS)* to verify the availability and status of the requested resources. If there are available resources, (III) the *Co-allocation Service* will select the best option based on the information from the *GIS*, and it will allocate the resources accordingly. At this point, the user ticket is associated with the reservation, which enables future interactions between the user and the system, allowing the user to verify the status of reservations, cancel a reservation or monitor the workflows. Subsequently, information pertaining to the allocated resources and the user ticket are stored in (IV) a *Data Base* (DB), hence enabling a checkpoint mechanism. This method is necessary in case a user wishes to access a previous workflow result or interact with the grid environment to submit or cancel workflows.

The ticket created by the authentication module might specify a duration time based on the time of the reservation. When a ticket is no longer valid, the resources reserved by the user are automatically released. When the user submits a workflow to the environment, the Reservation Service searches the information in the Database to verify the user ticket. Once the ticket is verified, the workflow is sent to the *Workflow Manager* (V).

Through the reservation module, users can plan the future use of resources based on their mobile requirements. Since the reservation is performed and monitored through their mobile device, the user does not need to worry about which device will make submissions and monitor workflows. Rather, they need to ensure that the resources have been previously allocated.

The development of the resource reservation module was enabled by GridSim [17]. The toolkit supports modeling and simulation of heterogeneous grid resources, users and application models. This toolkit will assist withthe implementing the Reservation Service component, modeling heterogeneous computational resources of variable performance, and testing the policy of advance resource reservation.

4 Experimental Results

Experiments were performed on the basis of the environmental configuration proposed in [3] and focus on the battery usage without concern about potential security threats. Therefore, the Java programming language was used for implementing the user-centric authentication service and integrating it with the other modules. In addition, the module present in the Mobile GUI, which is responsible for handling the user’s vCard and intercepting service requests, was implemented using the J2ME (Java 2 Micro Edition) Wireless Toolkit. In addition, the simulator GridSim was used as the grid environment.

The experimental environment consisted of a server containing the authentication service, which was integrated with the other modules presented in [3]. Also, the mobile devices used in our experiments were two Palm Tungsten C devices, each with a 400MHz processor, 64 MB RAM, built-in Wi-Fi (802.11b), and a Palm OS 5.2.1 operating system. As the integration of GPS with authentication system is still under development, for the current experiments we used a device without GPS.

Since the maintenance of battery life is one of the most critical and challenging problems found in mobile devices, such as PDAs and cell phones [18], new methods and techniques are required to reduce the dissipation of energy in such devices. Accordingly, we analyzed the efficiency of the proposed user-centric authentication mechanism based on the power consumption of mobile devices.

This analysis was performed by identifying a pre-defined sequence of ten service requests. Specifically, this analysis compared the execution of a sequence using the user-centric authentication approach and the traditional authentication approach. The traditional authentication approach refers to the device-centric authentication method used in [3], where a username and password are requested when the user moves from one mobile device to another. The device-centric authentication was chosen for analysis because it is one the most common authentication mechanisms in mobile grid environments, as indicated in the research. In order to evaluate the efficiency of the mechanism proposed in this paper, we simulated a user changing mobile devices. Therefore, the pre-defined requests were interspersed between the two devices by performing the pre-defined sequence in a mobile device, then running it in the other one.

Figure 6 presents empirical results using the battery consumption of the mobile devices as the metric for the performance of the proposed approach, which compared user-centric authentication to the traditional authentication. This evaluation was performed using the BatteryGraph software [19], which had been installed on the mobile devices. It provides the mean battery level, expressed in milivolts (mV), before and after the completion of each of the two applications.

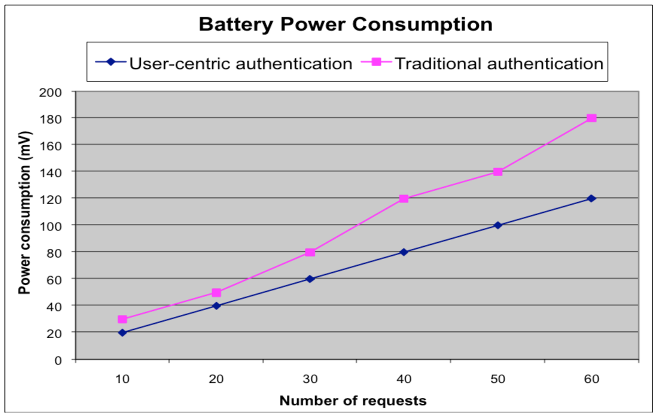


Fig. 6: Battery power consumption for two authentication approaches

Figure 6 indicates that the user-centric authentication approach demonstrates a consistent increase in power consumption based on the number of requests. In comparison, the traditional authentication approach causes a greater increase in the energy consumption of the battery. Thus, the proposed approach represents a significant reduction in the power consumption of the battery as compared to the traditional authentication mechanism.

The second experiment attempted to analyze the efficiency percentage of the proposed mechanism. Specifically, it consisted of running the same sequence of pre-defined requests several times until the battery was totally depleted. First, this procedure was performed using the traditional authentication approach and switching the mobile devices between the sets of requests. Subsequently, the same experiment was performed by executing the application with the user-centric authentication mechanism. As in the case of the traditional approach, this execution also used the process of interspersing the sets of requests between the two mobile devices.In addition, in order to acquire an accurate assessment, both experiments were performed three times.

Figure 7 presents the results obtained by using the two authentication approaches. The BatteryGraph was utilized for obtaining the percentage of the devices' charge level. Moreover, it also verifies the percentage of battery used during a time interval selected by the user.

As shown in Figure 7, the mechanism proposed in this work for user-centric authentication enables a greater autonomy of energy in comparison to the traditional authentication mechanism. For instance, the traditional authentication mechanism resulted in total battery exhaustion after 130 requests, while the proposed approach for user-centric authentication did not completely drain the battery until after 200 requests. Therefore, the user-centric authentication approach results in a noticeable increase of approximately 53% in the battery life.

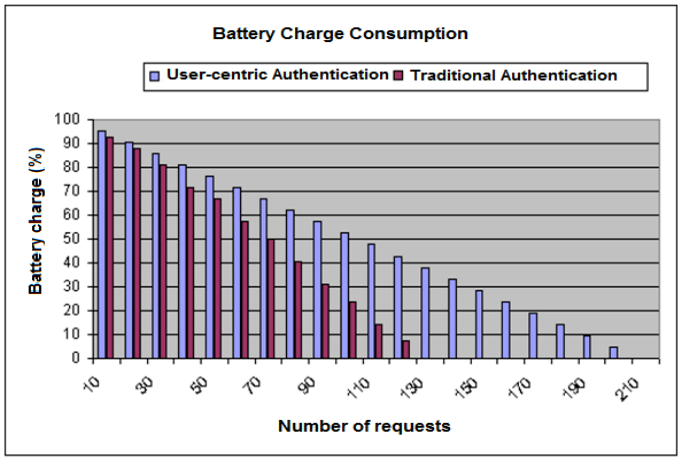


Fig. 7: Comparison of battery charge for two authentication approaches

5 Conclusions and Future Work

A user-centric authentication approach was proposed in this paper for enabling safe advanced resource reservation in mobile grid environments. The proposal addressed the shortcomings in [3], which were mainly due to the inefficiency of various components in the environment. This work aimed to create a safe and transparent system for users to submit tasks and reserve resources in mobile grid environments. Specifically, its primary objectives consisted of making the user's interaction with the environment more flexible and reducing the battery consumption of mobile devices, both of which were successfully achieved. In addition, the proposal also aimed to provide to the user with more efficient mobility resources in such environments.

As a future work we are planning to perform further simulations using other mobile device models in order to perform experiments regarding the user's location and the impact on the experience of the user. In addition, it is interesting to consider other important metrics in mobile grid environments, such as the occurrence of disconnections and interferences in the wireless network, user interactivity, and the ability of the system in detecting frauds. The latency of the user-centric authentication approach is important also to measure, particularly, the way in which it affects the resource reservation environments.

6 References

[1] I. Foster, "What is the Grid? A three Point Checklist", GridToday, vol.1, no.6, July, 2002.
[2] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations", International Journal of High Performance Computing Applications, vol. 15, no. 3, pp. 200-222, 2001.
[3] A. Rossetto, V. Borges, A. Silva, and M. Dantas, "SuMMIT – A framework for coordinating applications execution in mobile grid environments", Proceedings of the

8th IEEE/ACM International Conference on Grid Computing, pp. 129–136. IEEE Computer Society Washington, DC, USA, 2007

[4] D. Chu and M. Humphrey, "Mobile OGSI .NET: Grid computing on mobile devices", Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing, pp. 182–191. IEEE Computer Society Washington, DC, USA, 2004.

[5] A. Gomes, A. Ziviani, L. Lima, and M. Endler, "DICHOTOMY: A Resource Discovery and Scheduling Protocol for Multihop Ad hoc Mobile Grids", Proceedings of the 7th IEEE International Symposium on Cluster Computing and the Grid, pp. 719–724. IEEE Computer Society Washington, DC, USA, 2007.

[6] D. Saha and A. Mukherjee, "Pervasive computing: a paradigm for the 21st century", CIEEE Computer, vol. 36 no. 3, pp. 25–31, 2003.

[7] A. Takefusa, H. Nakada, T. Kudoh, Y. Tanaka, and S. Sekiguchi, "GridARS: An Advance Reservation-Based Grid Co-allocation Framework for Distributed Computing and Network Resources", Lecture Notes in Computer Science, 4942, pp. 152-168, 2008.

[8] M. Siddiqui, A. Villazon, R. Prodan, and T. Fahringer, "Advanced Reservation and Co-Allocation of Grid Resources: A Step towards an invisible Grid", Proceedings of 9th IEEE International Multitopic Conference, pp. 1–6. IEEE Computer Society Washington, DC, USA, 2005.

[9] T. Roblitz and A. Reinefeld, "Co-reservation with the concept of virtual resources", 5th IEEE International Symposium on Cluster Computing and the Grid, vol. 1, pp. 398–406. IEEE Computer Society Washington, DC, USA, 2005.

[10] I. Foster, C. Kesselman, C. Lee, B. Lindell, K. Nahrstedt, and A. Roy, "A distributed resource management architecture that supports advance reservations and co-allocation", 7th International Workshop on Quality of Service, pp. 27–36. IEEE Computer Society Washington, DC, USA, 1999.

[11] The WS-Resource Framework [Online]. Available: http://www.globus.org/wsrf/

[12] D. Mashima and M. Ahamad, "Towards an user-Centric Identity-Usage Monitoring System", Proceedings of the 3rd International Conference on Internet Monitoring and Protection, pp. 47–52. IEEE Computer Society Washington DC, USA, 2008.

[13] H. Choi and Y. Yi, "an user-Centric Privacy Authorization Model Based on Role and Session in the Context-Aware Home" Proceedings of the 8th IEEE International Conference on Computer and Information Technology Workshops, pp. 254–259, IEEE Computer Society Washington, DC, USA, 2008.

[14] A. Yamin, J. Barbosa, I. Augustin, L. da Silva, R. Real, C. Geyer, and G. Cavalheiro, "Towards merging context-aware, mobile and grid computing" International Journal of High Performance Computing Applications, vol. 17, no. 2, pp. 191-203, June, 2003.

[15] F. Dawson and T. Howes, "RFC2426: vCard MIME directory profile", RFC Editor, United States, 1998.

[16] M. Kulldorff et al. "SaTScan v7.0: Software for The Spatial and Space-Time Scan Statistics". Available: http://www.satscan.org/

[17] A. Sulistio and R. Buyya, "A Grid Simulation Infrastructure Supporting Advance Reservation", Proceedings of the 16th International Conference on Parallel and Distributed Computing and Systems, pp. 1-7, MIT, Cambridge, USA, 2004.

[18] S. Mohapatra, R. Cornea, H. Oh, K. Lee, M. Kim, N. Dutt, R. Gupta, A. Nicolau, S. Shukla, and N. Venkatasubramanian, "A Cross-Layer Approach for Power-Performance Optimization in Distributed Mobile Systems" Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, vol. 11, pp. 8. IEEE Computer Society Washington, DC, USA, 2005.

[19] BatteryGraph. Web Page. Available: http://palm.jeroenwitteman.com

Modeling the Risk of Failure in Grid Environments

Raid Alsoghayer, Karim Djemame
School of Computing, University of Leeds,
Leeds, LS2 9 JT, UK
{raid, karim}@comp.leeds.ac.uk

Abstract— *The need to model the risk of failure is critical to Grid resource providers. Moving from the best-effort approach in accepting Service Level Agreements (SLAs) to a risk-aware approach assists the Grid resource provider to offer a high level quality of service (QoS). In this paper we develop a mathematical model to predict the risk of failure of a Grid node using a discrete-time analytical model driven by reliability functions fitted to observed data. We consider four standard distributions Weibull, Gamma, Exponential, and Lognormal to fit the failure data. We find that the time between failures is well modeled by a Weibull distribution. We evaluate the model by comparing the predicted risk of failure with the observed risk of failure using availability data gathered from four Grid nodes. We find that the difference is not statistically significant between the observed risk of failure and the predicted risk of failure.*

Keywords: Availability Modeling, Grid computing, Risk of Failure prediction.

1. INTRODUCTION

Grid computing is the coordinated sharing of resources and solving problems in dynamic, multi-institutional virtual organizations. This sharing must be controlled with clear boundaries on what will be shared, who are allowed to share, and the conditions under which sharing occurs, whether the resources are hardware, software, or users [1-2]. The sharing should be done using standard, open, and general-purpose protocols and interfaces, and should deliver nontrivial quality of services (QoS) [3-4]. The sharing and coordination of resources on the Grid is complicated, since both the user and the Grid resource provider are geographically located in different time zones, and have competing needs. Therefore Service Level Agreements (SLAs) are used to either provide some measurable capability or perform a specific task. An SLA allows the user to know what is expected from a service without requiring detailed knowledge of the provider's policies [5-6].

Even with the introduction of SLAs commercial Grids are not attracting users and providers. Current Grid middleware (e.g. Globus Toolkit [7]) still follows the best-effort approach, there is a risk that users do not get any guarantees that their SLAs will not be violated. Also commercial resource providers are not attracted either. For a resource provider agreeing on an SLA without enough information about the state of resources and the availability of devices introduces a chance of violating the SLA, which results in a penalty fee. There is a risk attached to system failure, service unavailability, insufficient resources etc, which might lead to SLA violation. Without a method for assessing the risk of accepting an SLA providers are only able to make uncertain decisions regarding suitable SLA offers. Also end-users would like to know the risk of violating an SLA so that they can make decisions on what Grid resource provider to select and the acceptable cost/penalty fee. The risk-aware approach will also enable the Grid provider to understand the capacity of the infrastructure and plan future investment.

Significant work on risk assessment has been performed in the scope of the AssessGrid project [8]. The work has focused on qualitative and quantitative risk assessment and how it can be performed in a scenario where resource reservations are defined by a number of compute nodes (machines) required, the duration of the service, and scheduling constraints. The approach used in the AssessGrid project for failure estimations is based on the Possibility theory and assumed that Grid failure data are hardly available. On the other hand, our approach is to develop a mathematical model to predict the node risk of failure using a discrete-time analytical model driven by distribution functions fitted to observed data. We analyze failure data collected from four different nodes from two different Grid sites. We consider four standard distributions Weibull, Gamma, Exponential, and Lognormal to fit the failure data. We developed the risk estimation model using the observed distribution. In this paper a Grid site is a physical location containing Grid resources. A node is a computer running some type of Grid software. A site is build-up by one or more nodes. The node risk of failure at time *t* is the probability of the node not functioning at time *t.* This paper is organized as follows. The data collection process and the structure of the data records are presented in Section 2. In Section 3 we define the statistical distributions used throughout this research and describe our method for parameters estimation and goodness-of-fit test. In Section 4 we develop the mathematical model to predict the node risk of failure. To estimate the effectiveness of our model, we compare the model predicted risk of failure with the observed risk of failure in Section 5. Section 6 presents some related work. Then we conclude in section 7.

2. Data Collection

Research in the area of dependable computing depends on understanding how failures in the real world look like, e.g. knowledge of failure characteristics can be used in resource management to improve cluster availability [9]. Also creating realistic benchmarks and test-beds for reliability testing requires the knowledge of failure characteristics [10]. Therefore access to failure data is very important.

The Grid Operations Centre Data Base (GOCDB)[11] is a database of all sites within the Enabling Grids for E-science (EGEE)[12], the UK National Grid Service (NGS)[13] and Worldwide LHC Computing Grid (WLCG)[14] that contains information on sites, nodes, services, and downtime. GOCDB is publicly available and accessed following registration.

A downtime is a period of time for which a grid resource is declared to be inoperable. A downtime record contains unique downtime ID, downtime classification (scheduled or unscheduled), the severity of the downtime, the user who recorded the downtime, the date at which the downtime was added to GOCDB, the start and end of the downtime period, the description of the downtime, and the entity affected by the downtime.

Scheduled downtimes are planned and agreed in advance, while unscheduled downtimes are unplanned, usually triggered by an unexpected failure. EGEE define specific rules [15] about what should be classified as scheduled downtime and what should be classified as unscheduled downtime. The rules are based on the length of the intervention, the impact severity, and how long in advance the downtime is declared. Yet currently it is up to the person who declares the downtime to decide if it is scheduled or not.

The severity of the downtime is either “At Risk” (Resource will probably be working as normal, but may experience problems) or “Outage” (Resource will be completely unavailable).

The data collected in GOCDB is different compared to the data in error-logs. Error-logs are generated automatically and treat every unexpected event the same whether it resulted in a system failure or not. Also error-logs might contain multiple entries for the same event. On the other hand data in the GOCDB are created manually by system administrators. Human created failure data have two potential problems underreporting of failure events and misdiagnosing the cause of the downtime. While it is possible for a failure to being not reported at all, in this study we are assuming that this is not the case. Misdiagnosing the cause of the downtime is feasible. GOCDB does not have classification of the root cause (e.g. Hardware, Software, etc) it has only a description of what might cause the downtime. The diagnosis and description depend hugely on the administrators’ skills.

In this study we take into account the downtime data for two Grid Sites A and B from GOCDB. We considered two nodes from each site. The downtime data are for the whole node and show only the time when the node was down. The data span form July 2007 till January 2010. All data have scheduled and unscheduled downtime. Here we only consider unscheduled failures. The reason is that a resource provider uses advance reservation, which takes into account scheduled downtimes.

3. Fitting a Distribution to Failure Data

The time between failures on many computer systems have been observed to follow a Weibull distribution [9-10, 16-19]. Schroeder and Gibson [10] analyzed system failures collected over nine years from several high-performance computing systems at Los Alamos National Laboratories, and found the time between system failures was well-modelled by a Weibull distribution with shape parameter less than 1.

In this section we view the sequence of failure events for each node in the study as a stochastic process and we study the time between unscheduled failures, inter-arrival times. Even though times between failures on computer systems have been observed to follow a Weibull distribution, for the sake of investigating we fit the empirical Cumulative Density Function (CDF) at each node with four standard distributions Weibull, Gamma, Exponential, and Lognormal. We use maximum likelihood estimation (MLE) to estimate the distributions parameters and negative log-likelihood to test the goodness-of-fit.

Fig. 1 shows the empirical CDF at the 1st node site A, and Fig. 2 shows the empirical CDF at the 2nd node site A fitted by four standard distributions. Visual fit shows that the distribution between failures in both nodes is well modeled by a Weibull or Lognormal distribution, yet the Weibull is a better fit when tested using negative log-likelihood.

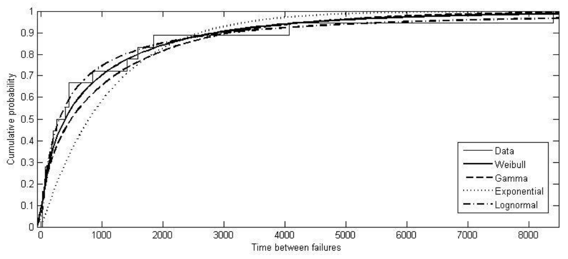


Fig. 1: CDF for inter-arrival times of failures Node 1_A.

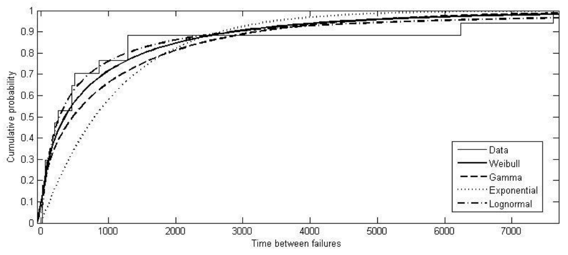


Fig. 2: CDF for inter-arrival times of failures Node 2_A.

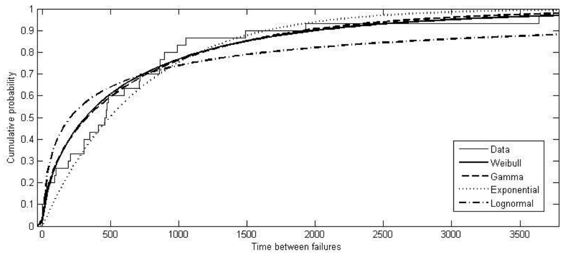


Fig. 3: CDF for inter-arrival times of failures Node 1_B.

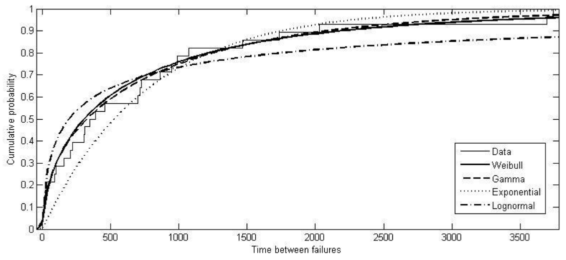


Fig. 4: CDF for inter-arrival times of failures Node 2_B.

Fig. 3 shows the empirical CDF at the 1st node site B, and Fig. 4 shows the empirical CDF at the 2nd node site B fitted by four standard distributions. Visual fit also shows that the distribution between failures in both nodes is well modeled by a Weibull or Gamma distribution. Both distributions create an equally good visual fit and the same negative log-likelihood.

This confirms the observation in previous researches that the time between failures is well-modeled by a Weibull distribution with shape parameter less than 1.

It is useful to know how the time since the last failure influences the expected time until the next failure. This notion is captured by a distribution's hazard rate function. An increasing hazard rate function predicts that if the time since a failure is long then the next failure is coming soon. And a decreasing hazard rate function predicts the reverse. The shape parameter of less than 1 indicates that the hazard rate function is decreasing, i.e. not seeing a failure for a long time decreases the chance of seeing one in the near future.

4. Developing the Risk Assessment Model

The risk of failure at time *t* of a node is the probability of the node not functioning at time *t*. This can be defined as one minus the probability of the node functioning at *t*. By computing the node availability *A(t)* we can compute the Risk of failure.

Risk of Failure = 1 – *A(t)*

To model the risk of failure for Grid nodes, we developed a three-state Markov model to represent the state of the node. State (0) is *UP*, which represents node is successfully operating. State (1) is *At Risk*, which represents a functioning node, but may experience problems. The final sate (2) is *Down*, which represent the node is unavailable and completely stops working. Fig. 5 shows a continuous time Markov model of the Grid node availability.

The node will start at state 0 and operate until either: (i) The performance degraded and the node transited to state 1. (ii) The node stops working and transited to state 2. $Z_W(t)$ is the rate of events that cause a node to transition from *Up* to *At Risk*, $Z_R(t)$ is the rate of recovery events that results in the node returning to the *Up* state, $Z_F(t)$ is the rate of events that leads to the node failure, and $Z_G(t)$ is the rate of repair events that results in the node return to *Up* state.

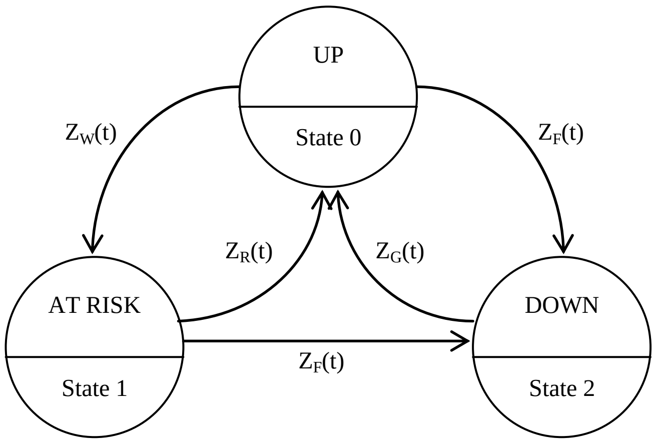


Fig. 5. Continuous-time Markov Model for Node Availability.

To determine the reliability functions for $Z_W(t)$, $Z_R(t)$, $Z_F(t)$, and $Z_G(t)$ for the continuous-time Markov model shown above we need to view the sequence of unscheduled events, in GODB, as a stochastic process. The data have two types of events, the first is At Risk, which represents the transition from state 0 to state 1, and the second is complete failure, which represents the transition from state 0 to state 2. The time to repair the system is the time to return it to state 0 from state 1 or 2. The time between events follow a Weibull distribution, therefore the reliability functions above are based on a Weibull probability density function, with unique shape α and scale λ values for each function.

$$Z_W(t) = \alpha_W \lambda_W \ (\lambda_W\ t)^{(\alpha_W - 1)}\ e^{-(\lambda_W\ t)^{\alpha_W}} \quad (1)$$

$$Z_R(t) = \alpha_R\ \lambda_R\ \ (\lambda_R\ t)^{(\alpha_R - 1)}\ e^{-(\lambda_R\ t)^{\alpha_R}} \quad (2)$$

$$Z_F(t) = \alpha_F\ \lambda_F\ \ (\lambda_F\ t)^{(\alpha_F - 1)}\ e^{-(\lambda_F\ t)^{\alpha_F}} \quad (3)$$

$$Z_G(t) = \alpha_G\ \lambda_G\ \ (\lambda_G\ t)^{(\alpha_G - 1)}\ e^{-(\lambda_G\ t)^{\alpha_G}} \quad (4)$$

Continuous-time Markov model are hard and complex to solve. Numerical integration techniques are one method of solving the model. An alternative method is to approximate the continuous-time process with discrete-time equivalents [20]. We will use the 2nd method because numerical integration involves some degree of approximation anyway. Fig. 6 shows the resulting discrete-time Markov model for time step Δt. Since more than one transition may occur during a time step, the model must take into account the joint probability of state transition.

The state transition probabilities for the discrete-time Markov model change over time, therefore we need to drive an expression for *A(n)*, *B(n)*, *C(n)*, *D(n)*, and *E(n)*. The model we drive is based on models developed by Howard [21] and Siewiorek and Swarz [20].

We are interested in calculating the probability transition equations, in which $q_{ij}(m, n)$ is the probability that the system is in state *j* at time *n* given that it was in state *i* at time *m* ($m \leq n$). With this notation, in matrix form the Chapman-Kolmogorov equation [21] is:

$$Q(m, n) = Q(m, k)\ Q(k, n) \quad m \leq k \leq n \quad (5)$$

Letting $k = n - 1$,

$$Q(m, n) = Q(m, n-1)\, Q(n-1, n) \tag{6}$$

Defining $P(n) = Q(n, n+1)$,

$$Q(m, n) = Q(m, n-1)P(n-1) \tag{7}$$

Expanding the equation recursively

$$\begin{aligned} Q(m, n) &= Q(m, n-2)\, P(n-2)\, P(n-1) \\ &= Q(m, n-3)\, P(n-3)\, P(n-2)\, P(n-1) \end{aligned} \tag{8}$$

Yielding the final solution

$$Q(m, n) = \prod_{i=m}^{n-1} P(i) \tag{9}$$

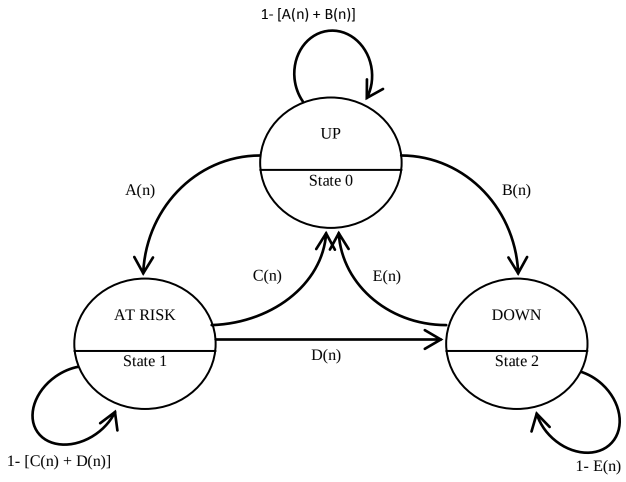


Fig. 6. Discrete-time Markov Model for Node Availability**.**

For converting from continuous-time probability functions to discrete-time probability function, a discrete-time probability distribution must be found that correspond to the continuous-time distribution. The corresponding parameters can then be calculated for the desired time-step Δt. Also a discrete-time approximation has to consider the probability of two failures during the same interval. The time-varying reliability functions $Z_W(t)$, $Z_R(t)$, $Z_F(t)$, and $Z_G(t)$ are based on a Weibull probability density function.

$$pdf = f(t) = \alpha\lambda(\lambda t)^{\alpha-1} e^{-(\lambda t)^{\alpha}} \tag{10}$$

The corresponding discrete Weibull function, probability mass function, is:

$$pmf = f(k) = q^{k^{\alpha}} - q^{(k+1)^{\alpha}} \tag{11}$$

Given that $f(k)$ is defined as the probability of an event occurring between time Δt and time $(k+1)\,\Delta t$ for some chosen interval size Δt. The probability mass function can be expressed as:

$$f(k) = P[\text{no event by } k\Delta t] - P[\text{no event by } (k+1)\Delta t] \tag{12}$$

$$f(k) = R(k) - R(k+1) \tag{13}$$

$R(k)$ is the reliability function. By substituting the continuous-time equivalents yields:

$$f(k) = R(k\Delta t) - R[(k+1)\,\Delta t] \tag{14}$$

$$f(k) = e^{-(\lambda k\,\Delta t)^{\alpha}} - e^{-[\lambda(k+1)\Delta t]^{\alpha}} \tag{15}$$

Rearranging terms we can find that

$$q = e^{-(\lambda\Delta t)^{\alpha}} \tag{16}$$

The probability mass functions $Z_W(n)$, $Z_R(n)$, $Z_F(n)$, and $Z_G(n)$ provide the reliability for a discrete time step $n = t_n/\Delta t$. The calculation of the transition probability functions for the discrete-time Markov model must take into account the joint probabilities of state transitions. The time varying functions are:

$$q_W = e^{-(\lambda_W\,\Delta t)^{\alpha_W}} \tag{17}$$

$$Z_W(n) = 1 - q_W^{\,(n+1)^{\alpha_W} - n^{\alpha_W}} \tag{18}$$

$$q_R = e^{-(\lambda_R\,\Delta t)^{\alpha_R}} \tag{19}$$

$$Z_R(n) = 1 - q_R^{\,(n+1)^{\alpha_R} - n^{\alpha_R}} \tag{20}$$

$$q_F = e^{-(\lambda_F\,\Delta t)^{\alpha_F}} \tag{21}$$

$$Z_F(n) = 1 - q_F^{\,(n+1)^{\alpha_F} - n^{\alpha_F}} \tag{22}$$

$$q_G = e^{-(\lambda_G\,\Delta t)^{\alpha_G}} \tag{23}$$

$$Z_G(n) = 1 - q_G^{\,(n+1)^{\alpha_G} - n^{\alpha_G}} \tag{24}$$

The transition probability functions in Fig. 2, which represent the probability of transition from one state to another state, are:

$$A(n) = [1 - Z_F(n)]\, Z_W(n) \tag{25}$$

$$B(n) = [1 - Z_W(n)]\, Z_F(n) \tag{26}$$

$$C(n) = [1 - Z_F(n)]\, Z_R(n) \tag{27}$$

$$D(n) = [1 - Z_R(n)]\, Z_F(n) \tag{28}$$

$$E(n) = Z_G(n) \tag{29}$$

The transition probability matrix $\mathbf{P}(n)$ =

$$\begin{bmatrix} 1 - [A(n) + B(n)] & A(n) & B(n) \\ C(n) & 1 - [C(n) + D(n)] & D(n) \\ E(n) & 0 & 1 - E(n) \end{bmatrix}$$

$A(n)$ is the probability of not going down and the probability of going from *Up* to *At Risk*, $B(n)$ is the probability of not going to at risk and the probability of going from *Up* to *Down*, $C(n)$ is the probability of not failing and going from *At Risk* to *Up*, $D(n)$ is the probability of not been recovered and going *Down*, and $E(n)$ is the probability of repairing the system and going from *Down* to *Up*.

The probability of transition $\mathbf{P}_{0,0}$ is the probability of remaining in state 0, which is 1 – the probability of leaving state 0, hence 1 – $[A(n) + B(n)]$. The same can be applied for the probability of transition $\mathbf{P}_{1,1}$ and $\mathbf{P}_{2,2}$.

$\mathbf{P}(n)$ can be used to compute instantaneous or point risk of failure which is the probability that the system will not be operational at any random time t. Yet the most important is the duration risk of failure which is the probability that the system will not be operational for the entire duration (e.g. job execution time). Computing duration risk of failure is an iterative process. Using appropriate values for α and λ, starting at T = start time, $\mathbf{P}(n)$ is computed forward for successive values of n until the desired finish time $t = n\,\Delta t$ is reached.

5. EXPERIMENTAL RESULTS

We take into account the down-time data for two nodes from Grid site A and two nodes from Grid site B. We compute the probability transition matrix **P**(n) for each node using the technique described in the previous section. We calculate the risk of failure as the sum of the probability of transitioning from *Up* to *At Risk* and the probability of transitioning from *Up* to *Down*. The observed risk of failure is the number of failures divided by the number of days in the time-span. We select the time-span to be 6 months for 2 reasons. (1) The data used to calculate the model span for years. (2) The Weibull shape parameter for Grid failures is less than 1 this means after a failure the risk of seeing one soon increased, therefore short time-span does not reflect the true behavior of the failures.

Fig. 7, 8, 9 and 10 shows the predicted risk of failure over a number of days and the observed risk of failure for node 1 site A, node 2 site A, node 1 site B, and node 2 site B correspondingly.

We use the Analysis of Variance (ANOVA) for testing the difference in the means between the predicted and the observed risk of failure. The test shows that the difference is considered to be not statistically significant with *P*= 0.508, *P*= 0.863, *P*= 0.232, and *P*= 0.088 respectively for the four experiments.

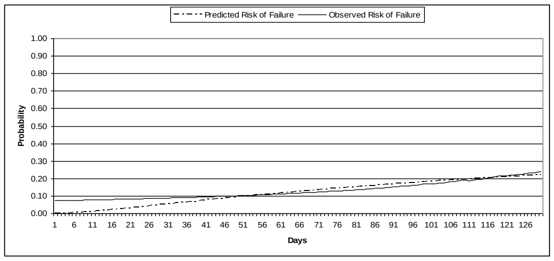


Fig. 7. Predicted & Observed Risk of Failure for Node 1_A.

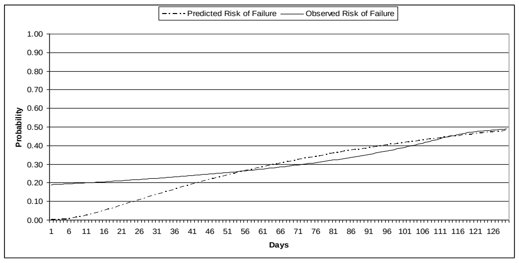


Fig. 8. Predicted & Observed Risk of Failure for Node 2_A.

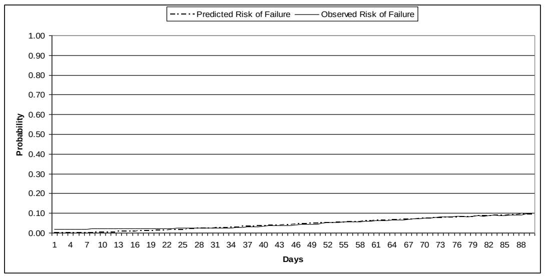


Fig. 9. Predicted & Observed Risk of Failure for Node 1_B.

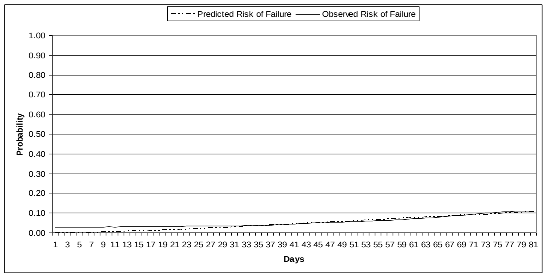


Fig. 10. Predicted & Observed Risk of Failure for Node 2_B.

Comparing Fig. 7, 8, 9 and 10, the most apparent feature is that the risk assessment model accurately predicts Grid node risk of failure. Therefore the Grid resource provider can integrate the risk assessment model to build the confidence in accepting SLAs.

The second observation is that nodes risk of failure of site A (Fig 7 and 8) is higher than nodes risk of failure of site B (Fig 9 and 10) The primary cause was the time to repair a failed node in each site. On site A the time to repair node 1 on average takes around 19 hours, while it takes around 25 hours for node 2. On site B the time to repair node 1 on average takes around 11 hours, while it takes around 9 hours for node 2.

6. RELATED WORK

Risk assessment in Grid computing has been addressed in the AssessGrid project [8, 22]. The main objective of the AssessGrid project is to address obstacles of a wide adoption of Grid computing by bringing risk management and assessment to this field, enabling use of Grid technologies in business and society. In this scope, AssessGrid delivers generic, customizable, trustworthy, and interoperable open-source software for risk assessment, risk management, and decision-support in Grids. The approach used to develop the risk assessment model in AssessGrid is different than the approach used in this research. The approach used in AssessGrid is based on the Possibility theory initiated by Zadeh in [23]. It assumed that Grid failure data are hardly available and are not frequent; therefore probability theory models cannot be used. Possibility theory is based on new concepts such as possibility measure, necessity measure, possibilistic distributions, etc. However this paper is the first attempt to compute the risk of failure using observed data from existing Grid systems.

Markov models are widely used to model System Availability and Reliability. [24] Investigated the use of Semi-Markov models to model node reliability on large supercomputing systems. [25] Used a two-phase cyclic non-homogeneous Markov chain to evaluate the performance of a replicated database. [26] Explored the use of homogeneous continuous time Markov chain with the amount of free memory to model the resource degradation of the computer system. [27] Studied the use of a cyclic non-homogeneous continuous time Markov chain to drive an optimal software rejuvenation model. Also a large number of studies that look at systems failure (or equivalently systems availability) is found in the literature and includes [9-10, 16-18, 28-29], to name a few. Most of these studies considered only short term availability data. Other studies used statistical modeling to predict resource failure (or availability) at Grid level not resources level. These studies only considered distribution fitting to the failure data. This approach does not take into account the effect of system repairs and also it only gave the probability of first failure at time *t*. Therefore if the system fails before time *t* then the estimated probability at time *t* is not accurate. However we develop a model that is based on the fitted distribution (Weibull) and takes into account the effect of system failures and repairs. The model estimate the probability of failure (Risk) at resource level taking into account the resources might fail at any time and require to get repaired.

7. Conclusion

The need to model risk of failure is critical to Grid providers. Moving from the best-effort approach in accepting SLAs to a risk-aware approach assists the Grid provider to offer a high level QoS. This increases the provider revenue, improves demand for resources, and decreases penalty fees to be paid in case of SLA violation. Also the reputation of the provider improves so that additional customers can be motivated to outsource part of their IT activities to the provider. In this paper we analyze the downtime data for two nodes from two Grid sites A and B. Our work shows that the time between failures in nodes is best fitted with a Weibull distribution with decreasing hazard rate, which is similar to other systems studied in the related work. We developed a mathematical model to predict the risk of failure using a discrete-time analytical model driven by distribution functions fitted to observed data. We evaluated the model by presenting both graphical and statistical evaluations. We found that the difference is not statistically significant between the observed risk and the predicted risk. In future work we plan to improve the risk assessment model by looking at other parameters affecting the risk of failure (e.g. system load, time of day, and availability of experts).

8. References

1. Foster, I. and C. Kesselman, *The Grid in a Nutshell*, in *Grid resource management : state of the art and future trends*, J. Nabrzyski, J.M. Schopf, and J. Weglarz, Editors. 2004, Kluwer Academic Publishers: Boston. p. 3-13.
2. Foster, I., C. Kesselman, and S. Tuecke, *The anatomy of the Grid*, in *Grid Computing : Making the Global Infrastructure a Reality*, F. Berman, G. Fox, and A.J.G. Hey, Editors. 2003, J. Wiley,: New York. p. 169-197.
3. Foster, I. and C. Kesselman, *Concepts and Architecture*, in *The grid : blueprint for a new computing infrastructure*, I. Foster and C. Kesselman, Editors. 2004, Morgan Kaufmann: Amsterdam ; Boston. p. 37-64.
4. Roure, D.D., et al., *The evolution of the Grid*, in *Grid Computing : Making the Global Infrastructure a Reality*, F. Berman, G. Fox, and A.J.G. Hey, Editors. 2003, J. Wiley,: New York. p. 65-100.
5. Czajkowski, K., et al., *Grid Service Level Agreements: Grid resource management with intermediaries* in *Grid resource management : state of the art and future trends*, J. Nabrzyski, J.M. Schopf, and J. Weglarz, Editors. 2004, Kluwer Academic Publishers: Boston. p. 119-134.
6. Czajkowski, K., et al., *SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems*. Lecture Notes in Computer Science. 2002. 153-183.
7. Foster, I. *Globus Toolkit Version 4: Software for Service-Oriented Systems*. in *FIP International Conference on Network and Parallel Computing*. 2005: Springer-Verlag LNCS 3779.
8. *The AssessGrid Project*. Available from: http://www.assessgrid.eu.
9. Taliver, H., P.M. Richard, and D.N. Thu, *Improving cluster availability using workstation validation.* SIGMETRICS Perform. Eval. Rev., 2002. **30**(1): p. 217-227.
10. Schroeder, B. and G.A. Gibson. *A large-scale study of failures in high-performance computing systems*. in *International Conference on Dependable Systems and Networks, DSN*. 2006.
11. *Grid Operations Centre DataBase*. Available from: http://www.grid-support.ac.uk/content/view/406/293/.
12. *Enabling Grids for E-sciencE* Available from: http://www.eu-egee.org/.
13. *National Grid Service* Available from: http://www.grid-support.ac.uk/.
14. *Worldwide LHC Computing Grid* Available from: http://lcg.web.cern.ch/LCG/.
15. *EGEE Production Infrastructure: Intervention Procedures*. Available from: https://edms.cern.ch/document/829986.
16. Lin, T.T.Y. and D.P. Siewiorek, *Error log analysis: statistical modeling and heuristic trend analysis.* Transactions on Reliability, IEEE, 1990. **39**(4): p. 419.
17. Jun, X., Z. Kalbarczyk, and R.K. Iyer. *Networked Windows NT system field failure data analysis*. in *Pacific Rim International Symposium on Dependable Computing*. 1999.
18. Iosup, A., et al. *On the dynamic resource availability in grids*. in *8th IEEE/ACM International Conference on Grid Computing* 2007.
19. Alsoghayer, R. and K. Djemame. *Probabilistic Risk Assessment for Resource Provision in Grids*. in *25th UK Performance Engineering Workshop*. 6-7 July 2009. School of Computing, University of Leeds.
20. Siewiorek, D.P. and R.S. Swarz, *Reliable computer systems : design and evaluation*. 3rd ed. 1998, Natick, Mass.: A K Peters.
21. Howard, R.A., *Dynamic probabilistic systems*. Vol. 1. 1971, New York ; London: Wiley.
22. Djemame, K., et al., *Introducing Risk Management into the Grid*, in *Proceedings of the Second IEEE International Conference on e-Science and Grid Computing*. 2006, IEEE Computer Society.
23. Zadeh, L.A., *Fuzzy sets as a basis for a theory of possibility.* Fuzzy Sets and Systems, 1999. **100**(Supplement 1): p. 9-34.
24. Hacker, T.J., F. Romero, and C.D. Carothers, *An analysis of clustered failures on large supercomputing systems.* Journal of Parallel and Distributed Computing, 2009. **69**(7): p. 652-665.
25. Platis, A., et al., *A Two-Phase Cyclic Nonhomogeneous Markov Chain Performability Evaluation by Explicit Approximate Inverses Applied to a Replicated Database System.* Journal of Mathematical Modelling and Algorithms, 2003. **2**(3): p. 235-249.
26. Koutras, V.P., A.N. Platis, and G.A. Gravvanis, *Software rejuvenation for resource optimization based on explicit approximate inverse preconditioning.* Applied Mathematics and Computation, 2007. **189**(1): p. 163-177.
27. Koutras, V.P., A.N. Platis, and G.A. Gravvanis, *On the optimization of free resources using non-homogeneous Markov chain software rejuvenation model.* Reliability Engineering & System Safety, 2007. **92**(12): p. 1724-1732.
28. Nadeem, F., R. Prodan, and T. Fahringer. *Characterizing, Modeling and Predicting Dynamic Resource Availability in a Large Scale Multi-purpose Grid*. in *8th IEEE International Symposium on Cluster Computing and the Grid, CCGRID '08.* 2008.
29. Nurmi, D., J. Brevik, and R. Wolski, *Modeling Machine Availability in Enterprise and Wide-Area Distributed Computing Environments*, in *Euro-Par 2005 Parallel Processing*. 2005, Springer Berlin / Heidelberg. p. 432-441.

Data Replication Strategy in the Data Grid

Leyli Mohammad Khanli[1], **Farzaneh Veghari Baheri**[2]
[1]Assistance Professor, Computer Science, University of Tabriz, Iran
[2]Islamic Azad University – Shabestar Branch, Tabriz, Iran

Abstract - *Data Grid is composed of storage resources. Data Grids are useful technique for management of the huge distributed and shared data resources efficiently. Most of the efforts in data management have been done on data replication. Data replication is a strategy to achieve better performance and reliability. Data replication generates multiple copies of the existing data to reduce access time. In this paper we improve the performance of data access and bandwidth consumption and reduce the access latency. In this paper, a dynamic data replication strategy is proposed, which is called Largest Access Largest Replication (LALR). We assume hierarchical architecture. This architecture has some clusters. LALR chooses a popular data and calculates a replica number. The data access history saved in the table in each cluster. In a case, if requested data does not available in a cluster then it will be send to coordinator. Coordinator finds appropriate data replication with the least cost. In order to does this, coordinator collect the history of each cluster then it determines number of replication for reduce the access latency. We simulate this algorithm to evaluate the performance of this dynamic replication strategy with other strategies in a cluster grid. The simulation results show that LALR has shortest data access time.*

Keywords: Cluster; Data Grid, Data Replication

1 Introduction

In data grids, many scientific and engineering applications require access to large amounts of distributed data (terabytes or petabytes) [11, 15]. The size and number of these data collections has been growing quickly in recent years. The data grid is a solution for the management of the huge distributed data resource. Essentially, the data grid is providing geographically distributed storage resource for distributed scientific and engineering applications. An important technique that speeds up data access in data grid is data replication. Data replication is an efficient method to achieve better performance and reliability by storing copies of data sets on different grid nodes. Creating replicas can reduce bandwidth consumption and cost of access in data grid systems. The read cost for send data over the data grid systems to the end-user is minimized with replication. The replication strategy is divided into two important subjects: which data should be replicated and how many replicas should be created [1, 2, 3, 4, 7, 14].

Several factors effect replica transfer times and execution performance in data grid. For example the few factor such as static and dynamic parameters are shown below [5, 6, 8, 13]:

- Static parameters: these parameters are not changing when the environment in data grid is changing such as the type and frequency of CPU and the speed of sending network card.
- Dynamic parameters: these parameters can change when the environment is changing. Such as CPU usage rate, band width. In this paper we focused on the dynamic parameters.

There are some methods for find number of replica, for example there are mathematical and cache methods. The mathematical method needs some fixed parameters. As you know, that parameters are not available the most time for us. The cache method has some problem. The problem is that the method cache is not compatible. But in this article, we use history of data access that it is optimum access time and it is very better answer than other method [9, 10, 12].

In this paper, a dynamic data replication algorithm is proposed and is called Largest Access Largest Replication (LALR). A hierarchical architecture of a data grid system is assumed. A cluster shows an organization unit which is a group of sits. We develop time access and reduce the number of replicate in grid environment. We suppose that requests enter through CH with a priority tag. There is a table for saving data-ID and the access-number and the service-time in every cluster. In every period of time the tables of clusters is collected in a table by a coordinator. Coordinator is a replica management that defined as a system that can create, identify, manage, and update replicas within the virtual organization. And then data type and the optimal number of replicas of a dataset are defined. We simulate our algorithm to evaluate the performance of this dynamic replication strategy. It is obtained the more optimum answer than the other methods. Providing the appropriate replication for one site will increase the performance of system. Summary, the time of data access and bandwidth consumption and fault tolerance is reduced.
Section 2, in this paper introduces some previous work on grid data replication. Section 3, explain our dynamic replication mechanism in detail and section 4 evaluate proposed algorithm. Finally, section 5 summarizes some results.

2 Related Works

The technique of data replication is not a new strategy. Data replication provides higher data availability, increased performance and lower bandwidth consumption. In some cases the application programs need different high volume data and this data may be on the different machines.

In order to get better access performance, they [9] use mathematical economic data replication with security. In their model, market demand is estimated then invalid data replications which are created by hackers are considered. Since economic problems are considered. For example if user pays money for their demands then suppliers should not give invalid data replication. Therefore should increase access performance in order to maximize the average profit. Since mathematical methods needs some parameter which are not available all the time. This model is not useful for any situation.

Authors of [10] have proposed fast parallel replication (FPR) tool which creates multiple distribution trees by pipelining point to point transfer and reduce the data replication time to multiple sites. In order to do that first a tree with widest shortest path is created by the use of Dijkstra's algorithm. Then the remaining trees are created with DFS and BFS approach. Through this method the time of replication will be minimized. But the tree method demands high capacity of memory.

Authors of [8] have proposed the multi-tier hierarchical data grid architecture. This algorithm proposed an efficient mechanism for sharing of data. Two dynamic replication algorithms, Simple Bottom-Up (SBU) and Aggregate Bottom-Up (ABU), are proposed in [8] for the multi-tier data grid. The basic idea of Simple Bottom-Up (SBU) is to create the replicas as close as possible to the users that request the files. If a requested file does not exist in the parent node of the user and if the parent node has enough available space, then that file replicates.

The basic idea of Aggregate Bottom-Up (ABU) is to aggregate the history records to the upper tier till it reaches the root. The ABU algorithm adds the number of accesses for the records the same parent node. The result record after aggregation, the parent node id will replace the node id for the same parent.

3 LALR dynamic replication algorithm

3.1 Proposed hierarchical architecture

In this paper, we assumed hierarchical architecture with the dynamic data replication mechanism. This architecture has some clusters. Proposed hierarchical architecture is shown in figure 1. In a hierarchical model there is a coordinator in the centre which is responsible for the following tasks:

- Making contact to send and receive messages
- Data transferring
- Storing and maintenance of data
- Ability of observation and following up requests
- Determining the type and number of replication

There is a table on the coordinator that is included following fields:

- **File_ID:** Data Identification
- **BW:** Bandwidth Low or High?
- **Replica_Cost:** Cost Of Create Replication
- **Priority:** Data Value High or Low?
- **Access_Number:** Data Access-Number
- **Service_time:** Long Of Time for Allocation Data
- **Transmission_Time_without_Replication**: Time of send in a case there is no replication
- **Value_Replicate:** Number of Replication
- **Transmission_Time_with_Replication:** Time of send in a case there is replication

The coordinator saves condition and cost of dataset. Long distance has high latency. Coordinator selects the nearest replica. Cost of communication is calculated by hop count.

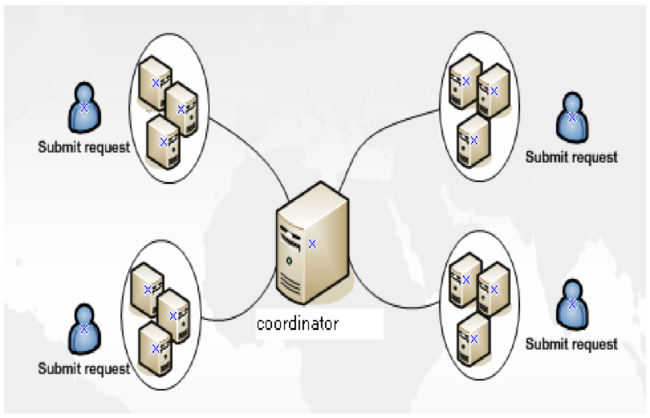


Figure 1. Proposed architectural hierarchical data replication

Proposed hierarchical architecture has some clusters. Each cluster consists of many relatively strong nodes which connected to each other.

There is a central node in every cluster named Cluster Head (CH) which is managing all nodes in cluster and monitors status of all nodes over the time as a central manager. In each cluster, only CH connected to internet and communicates with the entire clusters. There are some sits in each cluster. Each sites included thousands of nodes. We suppose that requests enter from CH with priority tag. There is a table on each CH in each cluster that is included following fields:

- **File_ID:** Data Identification
- **Data_Type:** Types can be local or remote
- **Access_Number:** Data Access-Number
- **Service_Time**: long of time for allocated data

Each cluster maintains a detailed record for each data. The format of a record in CH is <Data_ID, Data_Type, Access_Number, Service_Time>.

CH and coordinator are responsible for managing the access history. After summarizing the records by a CH, the information about number is sent to the coordinator. The number of data accesses should be aggregated for same data ID by coordinator. Therefore, there is a table in the coordinator for collecting the information. For example, if there is a record in the cluster A, <C, local, 10, 25> and in the cluster B, <C, local, 27, 30>, the result after collecting is <C, local, 37, 55>.

3.2 Largest Access Largest Replica (LALR) Algorithm

The Largest Access Largest Replication (LALR) is a dynamic replication algorithm which has two phases. First, finding which data is more popular. Second, we calculated how many replication need to be created.

In the first phase, list of requests are entered by users through the clusters and the time of request is registered as "Request_Time" and the time of allocation is registered as "Response_Time". Service_Time is available through expression (1):

$$Service\_Time = \mathrm{Re}\,sponce\_Time - \mathrm{Re}\,quest\_Time \qquad (1)$$

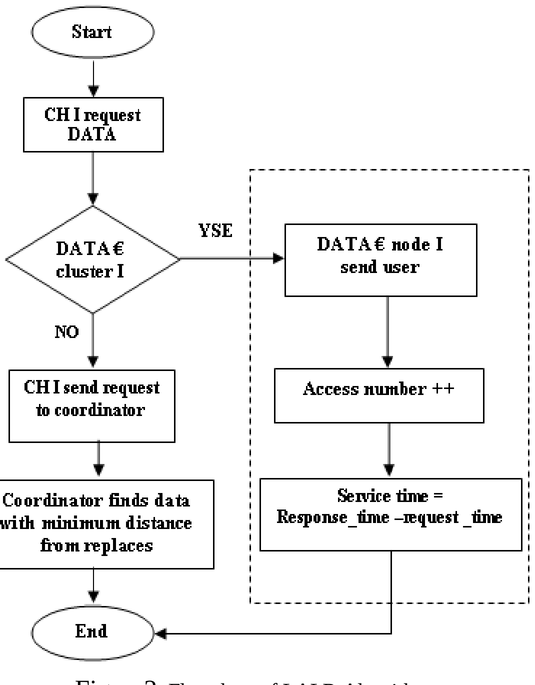

Start
CH I request DATA
DATA€ cluster I
YSE
NO
DATA€ node I send user
Access number ++
Service time = Response_time –request _time
CH I send request to coordinator
Coordinator finds data with minimum distance from replaces
End

Figure2. Flowchart of LALR Algorithm

If requested data is its own cluster then data will be allocated and this type of data is local, otherwise CH sends requested data to coordinator. Coordinator finds data with minimum distance and this data is remote for requested cluster.

When data is allocated, Access_Number in the cluster table is increased and Service_Time is calculated. This result is saving in the cluster table. These tables of clusters are updated in every access. Steps of updating the table are shown in figure 2.

Coordinator holds the location of data Replication. It should be mentioned that the long distance cause delay in receiving data so coordinator selects the data with the minimums distance and gets their identification. Relevant flowchart is shown in figure 2.

3.3 Calculation the type and the number of data replication

Coordinator collects the tables of clusters in each period of time (for example in the bank after the one month) and determines the number of replication with following expression:

$$Transmission\_Time = \frac{size\_of\_data}{BW} \qquad (2)$$

$$\mathrm{Re}\,plica\_Cost = Transmission\_Time \times Hope\_count$$
(3)

$$Value\_of\_\mathrm{Re}\,plication = \frac{Access\_Number \times \mathrm{Pr}\,iority \times Service\_Time}{Transmission\_Time}$$
(4)

For example these parameters in table 1 are as follows:

$$Transmission\_Time = \frac{size\_of\_data}{BW} = \frac{10000}{10} \qquad (2)$$

$$\mathrm{Re}\,plica\_Cost = 100 \times 1 = 100 \qquad (3)$$

$$Value\_of\_\mathrm{Re}\,plication = \frac{52 \times 3 \times 36}{1000} = 6 \qquad (4)$$

If the number of replication for a popular data in a cluster is less than the value of replication which is decided by the coordinator, then the number of replica will be copied. On the other hand, the LALR algorithm deleted the extra replication. If the copy of one data is not use at all, then it will be deleted by LALR.

4 Simulation

Simulation of this paper has been confirmed using the Visual Basic6 software beside the crystal report 7. At the first, we propose a hierarchical architecture with 4 clusters, A, B, C, D and each cluster has 3 sites. Simulation has been confirmed in 2 parts.

Part 1: we assume that requests have been saved and there is no replication. In each cluster there is only one data, because there are 4 clusters so we have 4 types of data. When data allocated, Access_Number in the cluster table is increased and Service_Time is calculated. These results are saving in the cluster tables following as table 1 for cluster A.

TABLE 1. CLUSTER A

File id	Type file	access number	Service time
1	Remote	12	21
2	Remote	20	15
3	Local	14	11
4	Remote	23	7

TABLE 2. CLUSTER B

File id	Type file	access number	Service time
1	Remote	20	4
2	Local	15	5
3	Remote	3	3
4	Remote	1	1

TABLE 5. TABLE OF COORDINATOR

File id	BW	Replica cost	Priority	Access number	Service time	Transmission time without replication	Value replicate	Transmission time with replication
1	Medium	Expensive	High	52	36	15	6	2
2	High	Medium	Low	43	29	3	2	15
3	Low	Cheap	Low	30	18	5	5	23
4	Medium	Cheap	High	57	17	4	3	2

TABLE 3. CLUSTER C

File id	Type file	access number	Service time
1	Remote	5	10
2	Remote	7	5
3	Remote	10	2
4	Local	33	1

TABLE 4. CLUSTER D

File id	Type file	access number	Service time
1	Local	15	1
2	Remote	1	4
3	Remote	3	2
4	Remote	0	8

These tables (table 2, table 3, and table 4) of clusters are updated in every access.

If each CH needs a data that it is not available in its own cluster, CH sends it for coordinator. After passing a period of time, the coordinator collects the tables of clusters as following table 5. The LALR algorithm is performed and the type and number of replication data is determined as expression 2, 3, and 4. In addition the total Service_Time is calculated for drawing charts and comparing them.

Part 2: Now the value of data replication is determined. Then the copies of data will be placed in a node with maximum requests. We restart running the simulation with 4 data after passing the same time period plus the services time and observe reduce of services time. All requests have priority label means that they have different value of important.

The chart of figure 3 shows the runtime of total entered requests in to the assumed environment in two states of no replication and with replication. Run time for replication mode in the LALR algorithm shows a reduction. The chart of figure 3 shows the runtime of total entered requests in to the assumed environment with 4 clusters in two states of no replication and with replication. Figure 3 shows LALR algorithm service time is lower by %25 compared to without replication.

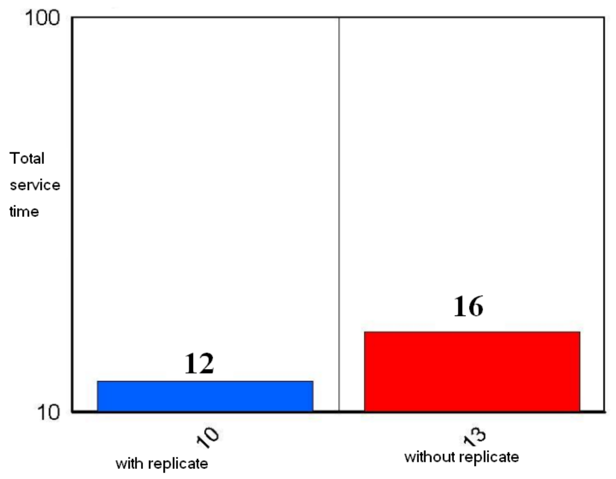


Figure 3: request time for 4 clusters

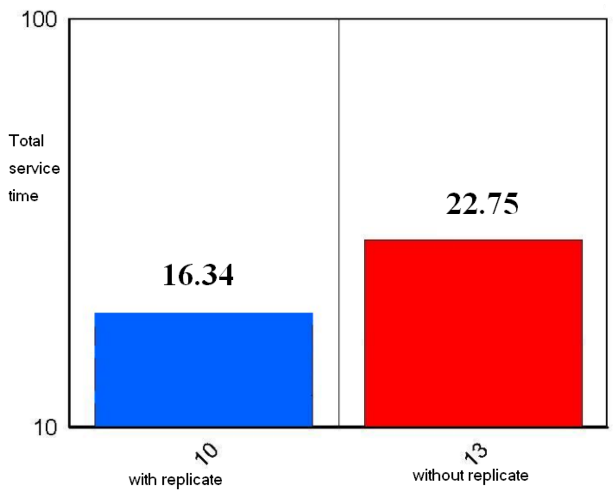


Figure 4: request time for 8 clusters

We assume that we have 8 clusters and 12 clusters which calculated access time for this assumption. The chart of figure 4 shows the runtime of total entered requests in to the assumed environment with 8 clusters in two states of no replication and with replication. Figure 4 shows LALR algorithm service time is lower by %28 compared to without replication.

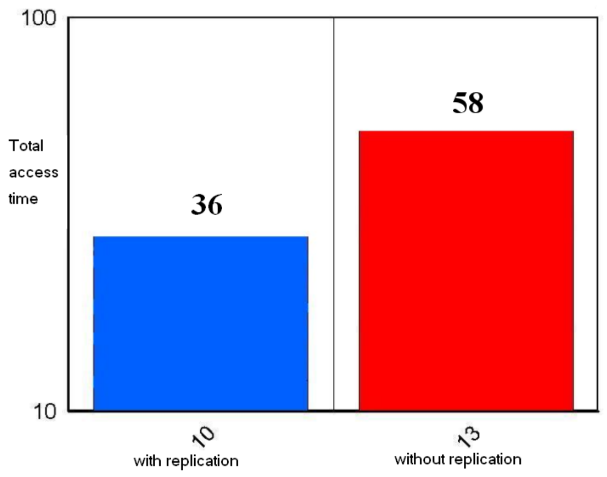


Figure 5: request time for 12 clusters

The chart of figure 5 shows the runtime of total entered requests in to the assumed environment with 8 clusters in two states of no replication and with replication. Figure 5 shows LALR algorithm is lower by %37 compared to without replication. The simulation results show that LALR improve the performance of data access and bandwidth consumption.

5 Conclusions

In this paper, we propose a dynamic replication strategy called Largest Access Largest Replication (LALR). At intervals, the dynamic replication algorithm gathers the data access history, which contains File_ID, Access_Number and the Service_Time that each request came from CH. Moreover, these history tables are collected by coordinator from each cluster.

According to access frequencies for all data that have been requested, popular data is found and a number of replication will be calculated. In order to evaluate the value of our dynamic replication strategy, we simulate a grid environment. All data access time was investigated in two with replication and without replication modes. The simulation of LALR algorithm shows that the total data access time is reducing.

6 References

[1] Y.Lin, J.Wu, P.Liu, "A List-Based Strategy for Optimal Replica Placement in Data Grid Systems", 37th International Conference on Parallel Processing IEEE, 2008

[2] Z.Jianfeng, Q.Leihua, Z.Dong, Z.Jinli, "A Duplicate-Aware Data Replication", 978-0-7695-3540-1/08 $25.00©2008 IEEE

[3] Y.Mansouri, R.Monsefi, "Optimal Number Of Replicas In Data Grid Environment ", 2007

[4] R.Chang, H.Chang, Y.wang, "A Dynamic Weighted Data Replication Strategy in Data Grids". 978-1-4244-1968-5/08/$25.00©2008 IEEE

[5] G.Nie, L.Zhang, Y.Liu, X.Zheng,Y.Shi, "Decision Analysis Of Data Mining Project Based On Bayesian Risk", Sciencedirect Elsevier Journal , 2009

[6] C.Yang, C. Huang, T.Hsiao, "A data grid file replication maintenance strategy using Bayesian networks" ,Eighth International Conference On Intelligent Systems Design And Applications IEEE , 2008

[7] R.Chang, J.Chang,S.Lin, "Job Scheduling And Data Replication On Data Grids", Future Generation Computer System 23, 2007

[8] M.Tang, B.Le, C.Yeo, X.Tang,"Dynamic Replication Algorithms for the Multi-Tier Data Grid", Future Generation Computer System 21, 2005

[9] J.Li,"A Replication Strategy in Data Grid Environment", IEEE 2009

[10] J.Li, "Fast Parallel File Replication in Data Grid ", 2009 IITA International Conference On Services Science Science, 2009 IEEE

[11] H.Lamehamedi, B.S.szymanski, Z.Shentu, "Data Replication Strategies in Grid Environments", Bejing, China, October 2002, IEEE Computer Society Press, Los Alamitos, Ca, 2002

[12] A.Domenici, F.Donno, G.Pucciani, H.Stockinger, K.Stockinger, "Replica Consistency in A Data Grid", Sciencedirect In 2004

[13] A.Chakrabarti, R.A.Dheepak, S.Sengupta," Integration of Scheduling and Replication in Data Grids", Springer- Verlag Berlin Heidelberg, 2004

[14] F.Carballeira, J.Carretero, A.Calderon, J.Daniel, M. Sanchez, "A Global and Parallel File System For Grids", Sciencedirect Future Generation Computer System 23, 2007

[15] T.Phan, K.Ranganathan, R.Sion ,"Evolving Toward the Perfect Schedule: Co-Scheduling Job Assignments and Data Replication in Wide-Area Systems Using A Genetic Algorithm" Springer-Verlag Berlin Heidelberg, 2005

Intelligent Replica Selection Strategy for Data Grid

Rafah M. Almuttairi, Rajeev Wankar, Atul Negi, C. R. Rao.
Department of Computer and Information Sciences, University of Hyderabad, Hyderabad, AP, 500046, India.

Abstract: *The timely availability of data is very crucial for the efficient execution of jobs in grids, especially in the context of data grids. In the replica selection procedure it is required to fetch data cached in replica across the grid in the least possible time. This selection problem has been investigated in literature thoroughly and most approaches minimize total execution time by minimizing delays in transferring accessed data. Usually this is done by minimizing lookup time by either using a classification technique like K-Nearest Neighbor rules [8] or using predictive techniques such as regression [12] or using neural network techniques [16]. All these methods attempt to predict the total transferring time instead of using traditional models which look up catalogs. In this paper, a totally new approach to replica selection is proposed that aims to optimize transfer by probing for current network congestion status and opts for the most efficient set of replica's sites that will work concurrently to transfer requested files or their parts. To achieve this goal, an association technique [3] is used to extract the replica's sites that have shown best possible efficiency and will work together to accelerate the response to the user's request. Our simulation results show an improvement of 29% as compared other reported work like the methods mentioned above and 40% better than traditional models that use replica lookup time.*

Keywords: Data Grid, Replica Selection Strategies, Association rules, Apriori Algorithm.

1 Introduction:

Scientific applications and researchers often require accessing, storing, transferring, analyzing, replicating and sharing huge amount of data in the domain of global climate change, high energy physics, and computational genomics. This generates large (on a terabytes scale) data volumes that are distributed in different locations around the world. To let the applications and researchers share and have an access to aforementioned files in a systematic way, we need an infrastructure which provides a secure, fast, reliable and transparent access. This infrastructure is provided by Data Grid which is optionally available in all grid infrastructures [2]. Data replication becomes important in data grids [1]. Data replication is a good technique that helps to move data by caching it at various nodes in data grid. The general idea of replication is to store copies of the same data in different locations so that data can be easily recovered if one copy at one location is lost. If data can be kept nearer (in terms of time required to access it) to the user, data access performance can be improved dramatically. Moreover, accessing data from a single location by all users is not feasible; it would lead to an increase in data access latency and single organization may not be able to handle such an enormous amount of data alone. Due to all these reasons data replication has become very important in data grids [1].

Since each file may have several replicas in distributed sites, to know where exactly these files are physically located, we use Replica Location Service (RLS) which has Local Replica Catalog (LRC) used to get physical file names and their location. Then, in order to reduce the network traffic and to enhance the overall execution performance of the job of the user/user application, the execution grid site must find the best replica site to get the requested file from it [7].

The decision of selecting the best replica site among many replicas sites is an important issue; because each site in grid has its own capabilities and characteristics such as: availability, security, network condition and cost, so the choosing process will affect the total execution performance especially data access latency [6].

Several attempts have been made in the past by many researchers to solve this important problem, but all of them have some merits and demerits. In this paper we present a new method for the solution for replica selection using Association rules of data mining which selects set of sites that has similar characteristics and then the method selects best among the selected sites.

The reminder of the paper is organized as follows. We present the related work in Section 2. Problem statement is declared in Section 3. The Data grid mining is briefly explained in Section 4. The replica selection strategies are discussed in Section 5. Our technique is explained in section 6. In Section 7 simulation input and results are presented. In section 8 we declared the comparison among all replica selection models. Discussion and conclusion is presented in Section 9 and the acknowledgment written in section 10.

2 Related works

Replica selection problem has been investigated by many researchers in the past. In 2001, Kavitha et al. [15], used traditional replica catalog based model, where for each new request Replica Location Service is queried to get the addresses of replica's sites and then probe the network link using Hop count method to select the best replica. This way of selection is not efficient because the number of hops does not reflect the actual network condition like Network Bandwidth and link's latency.

During 2001-2003, Sudharshan et al. [12,15,11] contributed many research results. In their work they used the history of previous file transfer information to predict the best site holing copy of requested file. When a file transfer has been made between two sites, the file size, the available network bandwidth, and transfer time are saved so it can be used later for training and testing the regression model to predict the actual transfer time. In their work they showed that data from various sources can help in better predictions than data from one source. They achieve a better accuracy in file transfer throughput prediction by using data from all of these three sources: data streams of network, file size, and past grid transfer information.

In 2005, Rashedur et al. [8], a replica selection technique called the K-Nearest Neighbor (KNN) rule is exploited to select the best replica from information gathered locally. The KNN rule selects the best replica for a file by considering previous file transfer logs indicating the history of the file and those nearby. This technique has a drawback as they mentioned in their paper: the misclassification will increased when for large file transfer and costs more than a couple of small file transfer misclassifications. Especially in the Gaussian random access pattern the accuracy is the lowest. Another drawback in KNN is that one needs to save all previous instances (requests of files) to use them to select the best replica site, which means it will take a time to search in the large history of data base and the result might or might not be correct.

In 2008, Rashedur et al. [16] proposed a predictive technique (NN based) to estimate the transfer time between sites. The predicted transfer time can be used as an estimate to select the best replica site among different sites. Simulation results demonstrate that Neural Network predictive technique works more accurately than the multi-regression model, which was used before NN [12, 15, 11]. However NN technique does not always give the right decision because the copy of the file may become no longer available (this is common in grid environment because the memory of site is limited) in the predicted site, so in this case Traditional Model will be used.

In 2009, A. Jaradat et al. [19] proposed a new approach that utilizes availability, security and time as selection criteria between different replicas, by adopting k-means clustering algorithm concepts to create a balanced (best) solution. The best site does not mean the site with shortest time of file transfer, but the site which has three accepted values: security level, availability and time of file transfer. Our work differs from the previous works by selecting not only one replica site, but number of sites that have similar characteristics in terms of stabilizing the network conditions. These sites concurrently work to send parts of a big file or different small files. So summarize, following are the drawbacks of the previous methods:

1- History file does not reflect the recent information; it is outdated (K-Nearest Neighbor rule).
2- Bandwidth alone and Hop counts alone do not describe the real network condition (Traditional Method, Neural Network and KNN).
3- In the classification method, the misclassification will increase in case of transferring large files and using a Gaussian Random file access pattern. (KNN)

None of the previous methods reflect the real picture of network links.

3 Problem statements:

To overcome the problems stated in the previous section, we propose a technique to measure the stability of network links before and during transfer time. In our work, by stable links we mean links with the least Standard Deviation of Single Trip Time (STT). STT is the time taken by the small packet to travel from replica's site to requester site. The STT delays include packet-transmission delays (the transmission rate out of each router and out of the replica site), packet-propagation delays (the propagation on each link), packet-queuing delays in intermediate routers and switches, and packet-processing delays (the processing delay at each router and at the replica site), therefore the delay from replica site to the requester site is the summation of all these delays [4]. To select the site which is stable we have to take all these delays into account. That is the major reason for using STT instead of Bandwidth and Hop count in our work.

Our technique has few features like:

1- Transfer large amount of data (Terabyte or above) from different sources simultaneously.
2- Allow a process to use multiple data streams to obtain data and biggest Maximum Transmission Unit.
3- Data consumers are allowed to get portions of data from different locations.
4- Work s properly with dynamic and static replica strategies.

4 Data Grid Mining

As we mentioned before our major work concern is to select the set of sites which has similar characteristic. To extract it, sites association rules of data mining approach is used. Apriori algorithm is the most popular algorithm used for the association rules discovery to extract the hidden knowledge of the large data base [5, 10].

5 Few Replica Selection Strategies:

When different sites hold replicas of a particular file, the access latency is minimized by selecting the best replica. In the following sub sections we will discuss few important replica section methods available in the literature.

5.1 Replica Selection by Traditional Model

In the Traditional Model (TM) of replica selection procedure whenever a new data grid job is submitted by resource broker to computing element, the computing element checks whether the files of the new job are available in the local storage element or not. If the files are locally available, the computing element immediately accesses them, otherwise, it gets the files from different distributed sites using following steps:

1- Consults the replica catalog with the logical file names of requested file to get the physical file names.
2- Probe the links between the requester site and replica sites using a testing route tools like Iperf [17]. The replica site with the highest network bandwidth or with the least Hop count measurement will be selected to get the file from. Replica Lookup Time is the time spent step one and two together [16].

5.2 Replica Selection by Neural Network (NN)

In there work the authors minimized the access latency by getting the files from the sites predicated by neural network instead of probing the routes as the TM does [16]. The drawbacks of this model are:

1- It works after 20 training sets of transferring file requests.
2- For each file request it contacts Local Replica Catalog with logical names of requested files to get the physical location names.
3- It minimizes the probing time by predicting the replica sites instead of probing network links.
4- It may or may not give a correct prediction. So the access latency is increased by using NN model whenever the prediction is incorrect.
5- In case of the correct prediction the selected site might or might not be the best site when we consider the congestion of the link between requester site and replica site.

5.3 Replica selection using K-Nearest Neighbor rule model (KNN)

In this model they have minimized the access latency by avoiding the considerable amount of replica lookup time [8]. In KNN model, when a new request for the file arrives all previous file requests (k requests) that are similar to the new one are considered to predict the best replica site. If the k-nearest rule model returns the site which has a copy of requested file, the classification is considered as a right classification; otherwise it is a wrong classification. The drawback of this model:

1- Needs to save all previous requests (File identifier, Time stamp, Replica Storage site's name) and search them all whenever a new request for the file arrives.
2- The selected site using KNN model may not have the requested file. In this case the access latency will increase because the model will go back to use the traditional model, means it doesn't work properly with dynamic replica strategy, the copy of the file may suddenly be deleted by replica site.
3- In Inter- Grid architecture, even though the prediction is correct the selected site might or might not be the best site with respect to the network conditions, at the time of file transfer.

We summarized characteristics of all studied models in the Table1:

Table 1. The limitations of all models

Replica's site satisfy:	TM	NN	KNN	EST
1- Shortest Time for trans. the file(s)	Y/N	Y/N	Y/N	Y
2- Avail. of file in the predictive site	Y	Y	Y/N	Y
3- Satisfy the Maximum Trans. Unit	N	N	N	Y
4- Satisfy Max. Number of data stream	N	N	N	Y
5- Size of replica affects efficiency of tech.	N	Y	Y	N

Followings are the limitations of previous replica selection models:

1- The replica site which is predicted by TM, NN and KNN may not achieve the least time to transfer the requested file.
2- The predicted site may not have the copy of the requested file any more.
3- None of the previously proposed methods took into account the maximum transmission unit of data packet.
4- The selection of predicted site is not done by taking into account for the maximum number of data streams that could be opened between two parties. (Our new technique takes all these points as criteria to select the best replicas.)
5- The size of requested file(s) affects the efficiency of selection techniques. In NN and KNN, the efficiency decreases when large files are requested, but there is no impact on the size of the file in case of TM and EST.

To overcome the drawbacks of all previous models, we propose a new technique to get the requested file(s) from the site(s) that require less time to transfer the file(s). To achieve this goal the network latency, maximum transmission unit and maximum number of data streams are considered as criteria for selecting the best set of replica's sites.

To reduce the total time to transfer the requested file(s) we use more than one replica sites to share the process of file(s) transfer. The selection of replica sites is done using following characteristics:

1- Uncongested links at the time of transferring the requested file(s) for parallel file transfer (Figure 1).
2- The biggest Maximum Transmission Units (MTU) (Section 6.1).
3- Maximum number of data streams can be opened between a computing site and a replica site (Section 6.2).

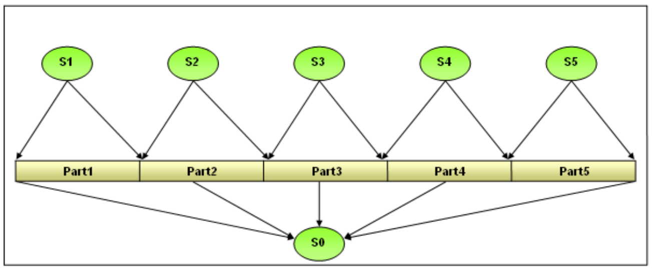


Figure1. Multiple sites send different parts of huge file

6 Efficient Set Technique (EST)

In this section a new Efficient Set Technique is proposed to solve a replica section problem in data grid. It removes many drawbacks which mentioned in section 3 of previously proposed methods. Followings are the steps of our method:

Step1: Receive requests from User/Resource Broker (RB).

Step2: Gather the replica location information from Replica Location Service (RLS).

Step3: Use *Iperf* service and probe the routes periodically between the executing site and all replica sites to build the Network History File, NHF, 2-D table where:

- Columns represent the replica sites(Let s be number sites); each column represents one site which has a copy of requested file.

- Rows represent the transactions (Let N be the number of transactions); one transaction represents the values of Single Trip Time (STTs), it is the latency of single trip starting from replicas sites to computing site (Requester site).

NHF=[$STT_{i,j}$] here i=1,2,..,N and j=1,2,…,s

Step 4: Calculate the threshold to convert the values of NHF to binary values and save it in a Binary Table called BT in the following way:

Consider block size as k consecutive transactions

-Calculate the mean:

$$M_{ij} = \sum_{s=i}^{i-k+1} STT_{sj} / k \quad \text{, for i=1,2,..,N-k}$$

-Calculate Standard Deviation:

$$S_{ij} = \sqrt{1/k \sum_{s=i}^{i+k-1} (STT_{s,j} - M_{ij})^2}$$

-Find $Q_{ij}=(S_{ij} / M_{ij})*100$

-Calculate the Average $Q_{ij.}$: $AQ_i = \sum_{j=1}^{s} Q_{ij} / s$

-Compare Qij with AQi and build BT

if ($Q_{ij} \geq AQ_i$) then $BT_{ij} = 1$ otherwise $BT_{ij} = 0$

Step5: Apply Association Model AM (Input (BT, C, S), Output (Ln)), where: n: is the order of the set for

C represents: Minimum confidence value.

S represents: Minimum support value

Ln= Union of Gnm over m for fixed n Group's order;

Step 6: Based on Maximum Transmission Unit and maximum number of data streams select a set Efficient Set in short ES , such that ES $\subseteq L_n$.

Step 7: Use any fast file transport services, (ex. GridFT[18]) to transport the file or part of it.

The ES obtained by using the above EST algorithm will be helpful in reducing execution time of a job in grid environment.

6.1 Maximum Transmission Unit

It is the largest physical packet size, measured in bytes that a network can transmit. To minimize the transferring time of requested files, the sites that have biggest MTU are chosen. MTU can be discovered by Iperf service [13] before starting the transmission of the files [4].

6.2 Maximum number of Data stream

It can further utilize the bandwidth provided by grid environment [4]. Whenever the numbers of streams are increased, the transmission efficiency is also increased. The number of streams can be set by users.

In our implementation, the numbers of streams are calculated using this formula [9]:

*Number of streams = bandwidth * delay / Window size.*

Using *Iperf* service [17] one can get the bandwidth, delay and Window size of Transport protocol for all replicas sites.

Before deploying any new replica selection strategy in the real grid environment it must be tested thoroughly to test the new replica selection strategy, analyze the results and compare it with the results of previous replica selection strategies. In our work we use a data grid simulator called OptorSim [13]. The next section presents the simulator OptorSim in brief, its architecture and the execution flows of the replication algorithms in the simulator.

7 Simulation:

To evaluate our approach we use a simulation package called OptorSim. OptorSim is a Data Grid simulator designed to allow experiments and evaluations of various replication optimization strategies in Grid environments

7.1 Architecture

The simulator design assumes that the Grid consists of a number of sites, each consisting of zero or more computing elements and zero or more storage elements. The computing elements provide computational resource and the storage elements serve as data storage resources for submitted jobs. A resource broker acts as a meta-scheduler that controls job scheduling for different computing elements. A job in OptorSim must access a set of files which may be located locally or at different storage sites.

7.2 Grid configuration Files in OptorSim.

Using OptorSim simulator our program gets input from the three configuration files:

7.2.1 The Grid Configuration File:

Being nodes of an intra-grid we encode the real life example of the PRAGMA Grid (Figure 2) into the simulator, therefore our grid configuration file will reflect the real network nodes and links condition of the PRAGMA Grid [20]. We select 19 sites of PRAGMA grid and we saved a copy of requested files in 9 sites which are: I= [S1, S3, S4, S5, S6, S7, S8, S14, S18]. S0 in the simulator is the executing site which represents (venus.uohyd.ernet.in) in PRAGMA real grid environment. S0 having the computing elements and enough memory to save requested files by submitted jobs J. J is list of jobs submitted to computing element of S0. J= [J1, J2, J3, J4, J5]. So in the simulator the grid configuration file is used to describe the topology of the participating sites, their associated network geometry and the content of each site shows resource availability, as shown in Figure 3. Each link between two sites shows the available network bandwidth which is expressed in Mbits/sec (M) or Gbits/sec (G). The circles referred to the sites which are separated by stars referred to routers.

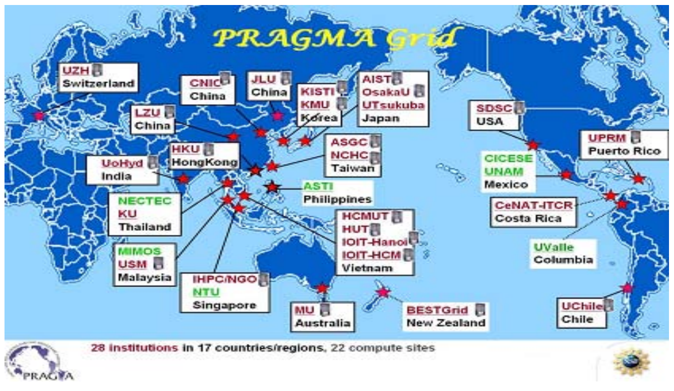


Figure 2. PRAGMA Grid, 28 institutions in 17 regions

7.2.2 Job Configuration File

This file contains names of jobs, list of required files for each job, a list of logical file names, their sizes in MB/GB and their unique numerical identifiers. It also contains job selection probability and schedule table with job execution time for each computing element.

7.2.3 Background Network Configuration File

The file used to describe background network traffic, it is *site-by-site matrix*, having for each pair of sites the name of data file containing the relevant STT information, the mean of STT and the standard deviation; keeping the source sites in the rows, and the destination sites in the columns. The last configuration file initializes different parameters for running the simulation. These parameters may include information such as total number of jobs to be run, file processing time, delays between each job submission, maximum queue size in each computing element, file access pattern, the optimization algorithm used, etc. As mentioned above, a job will typically request a set of logical filename(s) for data access. The order in which the files are requested is determined by the access pattern. In this simulation, we experimented with four access patterns: sequential (files are accessed in the order that has been stated in the job configuration file), random (files are accessed using flat random distribution), unitary random (file requests are one element away from previous file requests, but the direction will be random), and Gaussian random walk (files are accessed using a Gaussian distribution).

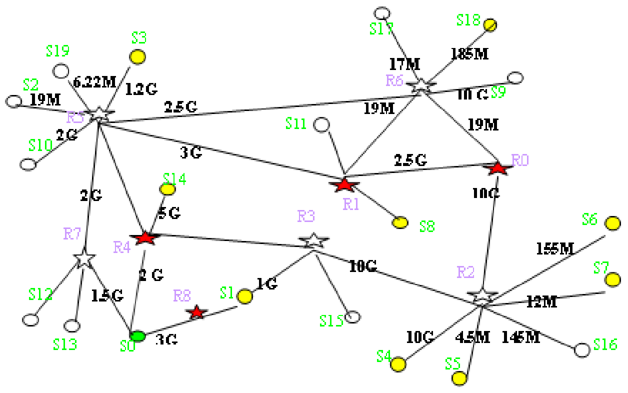


Figure 3.PRAGMA Grid and their associated network geometry

7.3 Execution replica selection in the Simulator:

We tested all replica selection models using same list of jobs and same environment to see the difference of the total time of executing jobs when we apply different models.

7.3.1 Using TM

If the file is in the local site, the computing element can access it instantly; otherwise the file must be transferred from the storage site that has the minimum transmit time to the requesting client. In the traditional model, the site gets the best replica of a file in the following way: it contacts the replica manager with a logical file name; the replica manager in turn calls the local replica catalog to determine physical locations of the logical file. Once it finds physical locations for the logical file, it gets the current network conditions (Bandwidth, number of Hop count, etc.), the replica site which has the maximum bandwidth or the least number of hop counts is considered as the best site to fetch the file from.

7.3.2 Using KNN

In this section we illustrate the execution of a computing element behavior using KNN. The computing element uses the traditional model for first thirty requests and stores: the file index, the destination site index number, the requesting site index number and the timestamp into a log file. When each site has enough transfer history, it requests the best site using the KNN rule; otherwise it contacts the replica catalog to find all file replicas as done in TM. This model works properly in two cases:

1- In the static replica strategy, a replica is presented until it's deleted by administrator or its time duration expired. In other words, the file is always remains available in replica site.
2- In the static network when there is no change in the network conditions.

7.3.3 Using transfer time prediction of NN and Regression model

In this model to train it, minimum 20 file transfer requests are required between two sites using traditional model. Parameters like the file size, the available network bandwidth, and transfer time are saved and can be used later for training and testing the neural network. Regression model is later used to predict the transfer time for new transfer file request.

We start training the NN with the first 20 training sets of data as implemented in [16]. Each set is presented to the neurons of the input layer and the whole network is trained with back-propagation algorithm so that the output neuron can predict the transfer time between the requesting site and the sites that hold replica of requested file. With this estimation on file transfer time, the computing element can request file from the site that has the lowest predicted transfer time. In our experiments this model works properly only when the bandwidth is the unique criteria for the network (which is not again a real condition in the Inter-Grid architecture). In general the latency is more effective parameter to chose the file from replica sites.

7.3.4 Using EST

In our proposed model we use criteria that if the computing element has sufficient scanning history of the routes of replica sites, it executes EST to find best replicas; otherwise it contacts the *Iperf* or *NWS* services to probe the related links and get a real view of routes. Initially, each computing element gets information from the replica catalog about the replica sites. In our simulation we assume a fixed replica lookup time, with (20 seconds). After the file transfer process is finished, the file index is recorded with the mean and standard deviation of STT into a log file NHF.

8 Comparisons:

In this section we compare the total time taken by file transfer process using different selections models:

8.1 Comparison between TM and EST:

The total transmission time for both of these methods is equal to *replica lookup time*. As we see in Figure 4, the EST is more efficient than traditional model. The main reason is that the EST selects the best replica from the sites with the stable links. In EST the retransmission is less than traditional method which selects the best replica from the sites having the least number of Hop Counts or the highest bandwidth which does not reflect the actual picture of the network, especially in the Inter-Grid architecture. Illustration of this is given in Figure 3. If the job J1 is submitting to S_0 it needs file resides in (S1, S3, S4, S14), using the traditional Model (with less number of Hop count) S_0 will ask S1 or S14 to deliver the file, because the number of Hops is 1, whereas the number of Hops between S_0 and S3 is 2 and between S_0 and S4 is 3. The time required to transfer the file in this case will be bigger than the time required from S3 because the two routers (R5 and R7) are uncongested whereas R4 is congested. Same thing will happen when TM chooses the best site depending on the highest bandwidth between two sites. EST chooses S3 to get the file because it has stable link with uncongested routers.

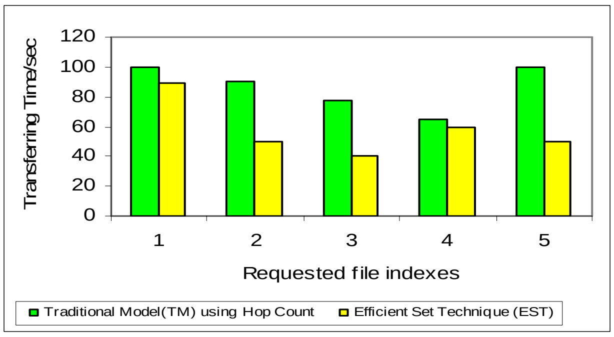


Figure 4. TM using Hop Count & EST

8-2 Comparison between NN and EST:

Both models have a Replica look up time. The NN needs history information (size of file and bandwidth) for at least 20 requested files, whereas EST needs only to probe the network links. As we see in Figure 5. The transferring time of requested files using neural network model takes more time than using EST because NN depends on two parameters which are: the size of requested file and the bandwidth between the requester site and replica site which holds a copy of requested file. Both of these parameters do not give a real picture of the network condition when the request for transfer has come. From Figure 5 it can be seen that EST performs better than NN in most of the cases because it depends on the actual picture of the links between sites.

8.3 Comparison between KNN and EST:

KNN model minimizes the total file transfer time by avoiding replica lookup time. It means that in this technique after the first thirty file requests, there is no need to contact the replica catalog or to probe the network bandwidth [16].

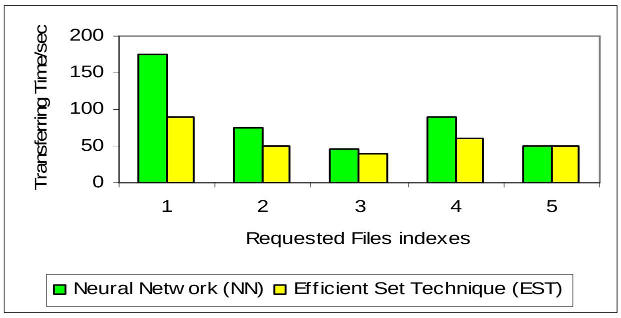
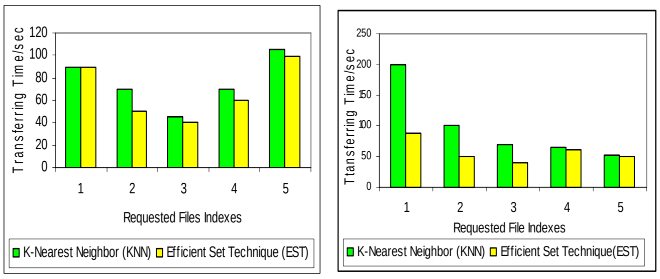


Figure 5. Transferring time using NN and EST

In the k-nearest neighbor rule, when a request for a file arrives, all previous data is considered to find a subset of previous file requests (k requests) that are similar to it and then it uses these to predict the best site holding copy of requested file. If the k-nearest rule returns the same site as returned by the TM, the file transfer is classified as the right classification; otherwise it is classified as the wrong classification.
In this section we compare the efficiency of the k-nearest rule with the EST. As we can see in the Figure 6, the KNN model even with right classification is less efficient than EST because the Grid environment is dynamic where user requests and network latencies vary significantly, therefore the site selected by the k-nearest rule may not be the best site for replica selection under the this condition.

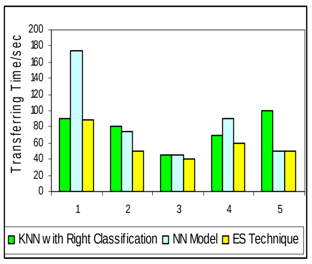
Due to sudden change in network conditions, the KNN rule may give wrong classifications. If the KNN gave five consecutive wrong classifications the simulator switches to traditional model again. We use k =5 for the k-nearest rule. Figures 6 (a) and 6 (b) depict the effect of right and wrong classifications on the total transferring time for different file requests and sequential access patterns.

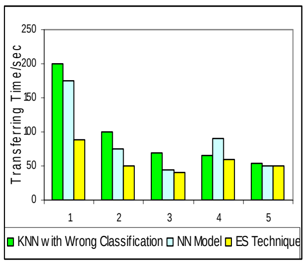
It can be seen that in Figure 6(b) the wrong classification makes the transferring time longer than the right one because it will be the summation of time spent to contact the predicted site which does not have the file any more and the replica lookup time. Figures 7(a) and 7(b) depict the different transferring time among three models (Traditional, KNN and EF), as we see the best technique is EST because it takes the file from the site that has a copy of the file, one that is stable and has maximum number of data streams which carry the data file packets.



(a) Right classification (b) Wrong Classification

Figure 6. Total transferring time using KNN with Right & Wrong classifications of replica's site vs. EST.





(a) Right classification (b) Wrong Classification

Figure 7. Total transferring time using KNN vs. NN & EST.

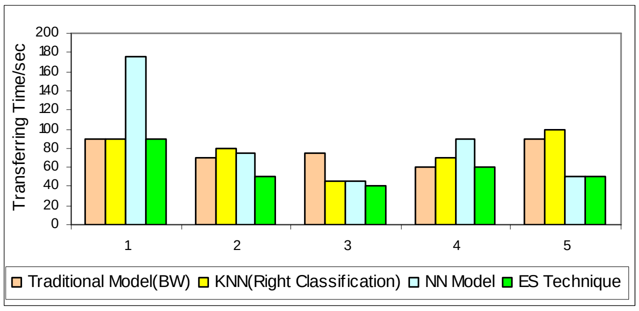


Figure 8. Compare total transferring time using all selection techniques.

9 Discussion and conclusion

In this paper we presented a novel method known as Efficient Set Technique (EST) to choose the best set of replicas in a dynamic and static replica strategies of data grid environment. We compared this method with the methods available in the literature for solving the same problem. Followings are the observations:

1- Our technique, EST works properly with two types of replica strategies: static and dynamic whereas most of the other models work with static better than dynamic replica strategy.

2- Our technique will select a set of sites having similar characteristics at the point of file transfer, whereas others will select one site as a best replica's site. In case if these methods also apply the same concept of having more than one site, perhaps the same result will not be achieved.

3- In previous methods, the selected replica site may or may not have the requested file since they depend on history of file requests which may be outdated information. Whereas in our method we do not have such a problem since we depend on present information of Local Replica Catalog.

4- Some models like traditional method depends upon the Bandwidth alone or Hop counts alone which do not describe the real network condition, whereas we depend on the STT which reflects the real network conditions.

10 Acknowledgements

Authors wish to register their sincere thanks to Prof. Arun Agarwal, from GridLabs Department of Computer and Information Sciences, University of Hyderabad, India for providing all the infrastructural and computational support required to carry out this work. His academic suggestions to improve the quality of the work are also highly appreciated/ acknowledged.

References

[1] Lei, M., S.V. Vrbsky and Q. Zijie, 2007, Online grid replication optimizers to improve system reliability, IPDPS 2007, March 26-30, DCSc, Alabama University.

[2] Ahmar Abbas, Grid Computing: A Practical Guide to Technology and Applications, 2006.

[3] Jochen Hipp, Ulrich Güntzer, and Gholamreza Nakhaeizadeh, Algorithms for association rule mining - SIGKDD Explorations, 2(2):1-58, 2000.

[4] James F. Kurose, Keith W. Ross, Computer Networking A Top-Down Approach Featuring the Internet, third edition.

[5] Agrawal R, Srikant R. Fast algorithms for mining association rules In, Proc. 20th VLDB Conference, 1994.

[6] H. Hamad E. AL-Mistarihi and C. H. Y., Response,Time Optimization for Replica Selection Service in Data Grids, USM, Pulau Pinang, Malaysia, pp 487-493, 2008.

[7] Francesco Palmieri and Silvio Pardi, Network-Aware Replica Optimization in the SCoPE Grid, Italy, 2008.

[8] Rashedur M. Rahman, Ken Barker, Reda Alhajj, Replica selection in grid env. data-mining approach, (DSGC),pp: 695 – 700 , 2005

[9] W. Zhang, W. Tong, Z. Chen and R. Glowinski, Intelligent File Transfer Protocol for Grid Environment, International Conference on HPCA, August 8–10, 2004.

[10] A. K. Pujari, Data Mining Techniques. Niversities Press, India, 2001.

[11] S. Vazhkudai, S Tuecke, I. Foster, Replica selection in the globus data grid, in: First IEEE/ACM International Conference on Cluster Computing and the Grid, CCGrid 2001.

[12] S. Vazhkudai, J. Schopf, Using regression techniques to predict large data transfers, in: Computing: Infrastructure and Applications, The International Journal of High Performance Computing Applications, IJHPCA , August, 2003.

[13] W. Bell, et al., OptorSim _ A grid simulator for studying dynamic data replication strategies, Journal of HPCA, 17 (4) (2003).

[14] http://www.javaworld.com/javaworld/jw-02-2009/jw-02-servlet3.html

[15] S. Vazhkudai, J. Schopf, I. Foster, Predicting the performance of wide-area data transfers, in: 16th International PDPS, 2002.

[16] RM. Rahman, K Barker and R Alhajj, Replica selection strategies in data grid, Journal of Parallel and Distributed Computing, Volume 68, Issue 12, Pages 1561-1574, December 2008.

[17] A. Tirumala, J. Ferguson, Iperf 1.2 - The TCP/UDP BW Measurement Tool 02.

[18] R.M. Rahman, K. Barker, R. Alhajj, Predicting the performance of GridFTP transfers,USA ,2003.

[19] A. Jaradat, R. Salleh and A. Abid, Imitating K-Means to Enhance Data Selection, Journal of Applied Sciences 9 (19): 3569-3574, 2009, ISSN 1812-5654, Asian Network for Scientific Information-2009.

[20] http://goc.pragma-grid.net/pragmadoc.

Aglets Mobile Agent Based Grid Monitoring System

Arindam Choudhury, Inderveer Chana
Computer Science and Engineering Department, Thapar University, Patiala, Punjab, India

Abstract – *Grid computing is dynamic. Any resource can leave or join the system at any time. In grid, mechanisms are needed to monitor available resources. Monitoring is also required for load-balancing and fault-tolerance. Fault is inevitable in grid as its supports various technologies and heterogeneous resources. In this paper a mobile agent based monitoring system for proposed for grid environment. The proposed system is designed using Aglets mobile agent framework.*

Keywords: Grid Computing, Aglets Mobile Agent, Monitoring System.

1 Introduction

Grid computing [1] concentrates on large scale resource sharing to provide solution to the problems requiring large storage and processing. Resources shared are spanned among different administrative domains and they are highly heterogeneous in nature. Grid computing is concerned with efficient utilization of these heterogeneous systems with efficient workload management. Resources can be compute cycles, storage system, network bandwidth, memory space, data transfer, and simulation etc.

To support interaction between heterogeneous resources from different administrative domain sharing must be highly controlled and done with consensus of both resource consumers and resource providers. Grid resource management [2] [3] system establishes a mutual agreement between a resource provider and resource consumer to perform some task on behalf of the consumer. It also manages available resources and system workloads accordingly to uniform utilization of resources of grid. It discovers the resources joining or leaving the grid dynamically. Grid resource management manages how the resources are allocated, assigned, authenticated, authorized, assured, accounted, and audited.

Grid has to deal with various technologies and protocols to accommodate heterogeneous resources form different Virtual Organizations [4]. Interaction between these technologies, protocols and systems may result into failures. To achieve high-performance in grid, it is critical to monitor and manage the system. Monitoring data helps to determine the source of the performance problems, fault detection and recovery. Monitoring systems also helps in load balancing.

Grid is not a controlled environment. Any resource can leave or join the environment at any time. Moreover conflict between policies, interaction between resources of different administrative domains may result in failures. Besides of traditional hardware faults, software faults and network faults grid suffers from two more types of failures [5]: Interaction faults which caused by protocol incompatibilities, security incompatibilities, policy problems, timing overhead etc. and omission faults. Faults in grid can be mitigated using retrying, replication and checkpointing mechanism.

Monitoring [6] [7] is crucial in a variety of cases such as scheduling, data replication, accounting, performance analysis and optimization of distributed systems or individual systems, self-tuning applications and many more. It also provides real time monitoring for the availability of resources and their utilization. This information is effective for effective management of resources. It also helps to detect faults and bottlenecks.

Monitoring system provides information about system configuration, network configuration, installed operating system, CPU information, free memory available, and load average etc of available grid nodes. This information helps in load balancing the grid and choosing best node to submit job.

Monitoring system also helps in fault-tolerance by providing information about the source of the problem. The job encountering failure can be submitted to a suitable node using the monitoring system.

There are many monitoring system available for monitoring grid environment such as Autopilot [8], Hawkeye [9], Mercury, NetLogger [10], and RGMA etc.

The purpose of the paper is to propose a mobile agent based monitoring system for grid computing. Software mobile agent [14] is a new computing paradigm where the client and server are merged to become host to provide high degree of flexibility. In this paradigm code is sent to the data which minimizes the use of network bandwidth. Mobile agents are a optimum choice to develop a monitoring system for grid computing as they reduce network load, execute asynchronously and autonomously, naturally heterogeneous, robust and fault-tolerant.

The rest of paper is organized as follows: Section 2 provides discussion about various technologies used in the proposed system. Section 3 discusses about design and deployment details of the monitoring system. It also provides the experimental results of the monitoring system. Finally section 4 concludes this paper.

The following sub-section discusses various technologies and programming paradigm used in the proposed monitoring system.

2 Technologies Used

In the proposed system java based mobile agent framework aglets is used. Hyperic SIGAR is used to gather system information locally on each system. MySQL database system is used to maintain the monitoring data. The website is designed using PHP and hosted on local network using apache HTTP server.

2.1.1 Globus Toolkit

Globus Toolkit [11] 4.2.1 is used to create grid for deploying and testing the proposed monitoring system. Globus Toolkit is developed and provided by Globus Alliance. Globus Toolkit includes software and libraries for resource monitoring, resource management, data management, communication, fault detection, security and portability. GT4 uses Web services mechanism heavily to provide flexible, extensible, and widely adopted XML-based mechanisms for describing, discovering, and invoking network services. Globus Toolkit is open source, well documented and mailing-list is available for users query.

2.1.2 Hyperic SIGAR

Hyperic System Information Gatherer (SIGAR) [12] is an open source cross platform API. SIGAR supports Linux, Windows, FreeBSD, Solaris etc operating system and architecture. SIGAR API gives portable access to inventory and monitoring data including:

- System memory, swap, CPU, load average, uptime, logins.
- Per-process memory, CPU, credential info, state, arguments, environment, open files.
- File system detection and metrics.
- Network interface detection, configuration information and metrics.
- Network route and connection tables.

Hyperic SIGAR is operated from the command-line tools. Hyperic SIGAR has been used in implementation as it is easy to use, open source and well documented.

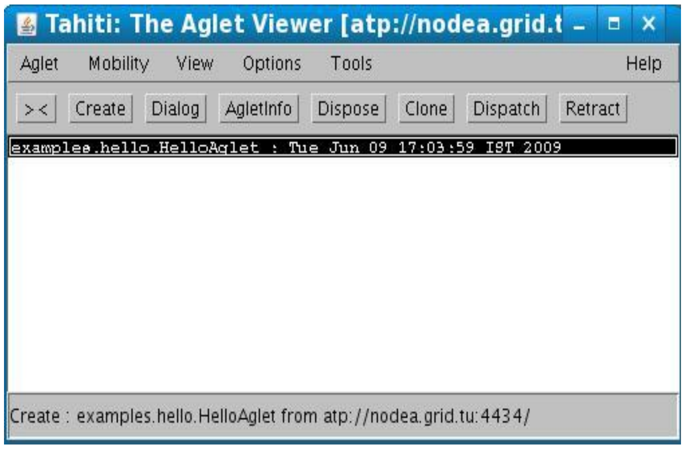
2.1.3 Aglets

Aglets [13] is a java based mobile agent platform. An aglet is a java mobile agent which can autonomously and spontaneously move from a system to another. An aglet is simply an object on which a thread executes on. This approach is very similar to the applets or servlets. Aglets has been used in the proposed monitoring system as it is open source and well documented. Aglets can migrate in both directions (in and out the platform) and message between local and migrated agents. Aglets supports weak mobility, only code is needed to transport without any particular information about the execution of the code, and aglets mobile agents are restarted from the entry point of the code after migration to other machines. To overcome these limitation aglets is designed to keep its java state after migration. The aglets code will restart from an entry point but no re-initialization will happen. Proxy is used in aglets for communication without hampering the security.

Aglets mobile agent platform is composed of following three parts:

- Aglet mobile agent platform: is the core platform, able to manage the agents.
- Tahiti: the server in-charge of managing the mobility of the agents. It comes with
- A graphical user interface.

Aglets software development kit: is a library that provides developers all the facilities required to write mobile agents complaint to the Aglets MAP.



```
examples.hello.HelloAglet : Tue Jun 09 17:03:59 IST 2009
Create : examples.hello.HelloAglet from atp://nodea.grid.tu:4434/
```

Figure 1 Tahiti Server

Figure 1 shows the Tahiti server, it provides functions to manage aglets mobile agents. The basic functions are:

- Create: allow administrators to create new agent instances.
- Dialog: sends message to the selected agent.
- AgletInfo: opens dialog window with information about the selected agent.

- Dispose: allow administrator to kill a running agent.
- Clone: allow administrator to create an identical copy of running agent.
- Dispatch: allow administrator to order an agent to migrate to another aglets platform.
- Retract: allow administrator to order an agent to come back from a remote aglets platform.

Aglets has been used in the implementation for its ease of use, functionalities and available documentation.

The next section discusses the topology, architecture, design of the mobile agent based monitoring system. It also details the experimental results.

3 Design of the Proposed System

This section discusses about the design of monitoring system. It describes the topology and architecture of the proposed monitoring system and the design of agents.

3.1.1 Topology of the Monitoring System

The topology of the monitoring system is shown below:

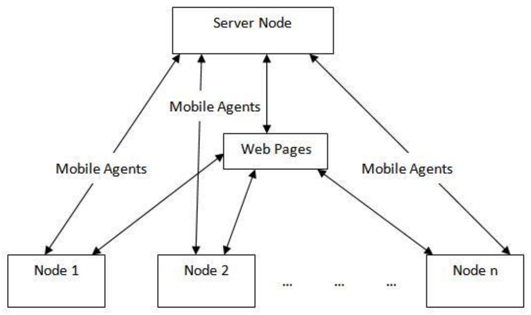


Figure 2 Topology of the monitoring system

Server node sends mobile agents to the other nodes of the network. These mobile agents gather resource information from the nodes. Then they return to the server node and update the database. Web pages are used to provide these resources information to users and hosted on local network using apache HTTP server. Any system on local network running apache HTTP server can access these web pages.

3.1.2 Architecture of the Monitoring System

This section discusses about the architecture of mobile agent based monitoring system. Figure 3 presents a pictorial view of the system.

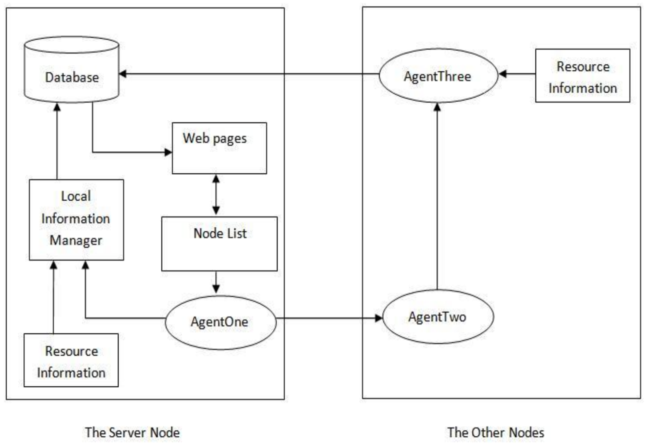


Figure 3 Architecture of the monitoring system

The system uses Hyperic SIGAR to gather information about the resources of local system. This resource information is managed, processed and updated to a database. The server's local resource information and remote system's resource information is collected differently. Local Information Manager is used to collect and process the local resource information and updates the database, for remote nodes mobile agents are used to collect and process the resource information. Local Information Manager is a Java file that reads the resource information generated by Hyperic SIGAR and updates it to the database. To manage resource information of remote systems three mobile agents are used. These mobile agents are: AgentOne, AgentTwo, and AgentThree. AgentOne invokes the local information manager on server. Agents are needed to dispatch to the remote nodes that are needed to be monitored. Addresses of these nodes are stored in a file which is denoted as "Node List" in the Figure 3. The detailed discussion of agents behavior are followed.

3.1.3 Design of Mobile Agents

Three mobile agents are used in the monitoring system as already discussed. These agents are AgentOne, AgentTwo, and AgentThree. Users need to feed AgentOne to the Tahiti Server; users also need to dispose the AgentOne manually. AgentTwo and AgentThree are generated and disposed automatically.

Next subsections give detailed description of working of AgentOne, AgentTwo and AgentThree. These mobile agents are discussed using work flow diagrams.

a) Design of AgentOne

AgentOne starts the monitoring process. Figure 4 shows the workflow of AgentOne.

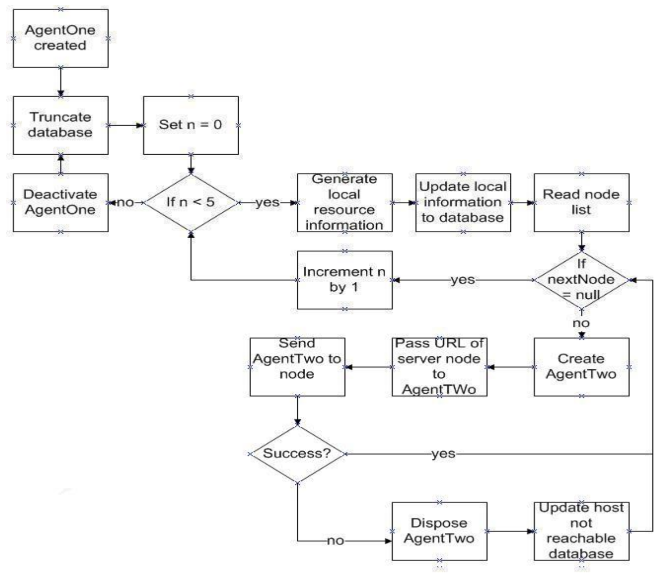


Figure 4 workflow of AgentOne

AgentOne truncates the monitoring systems database to delete all the previous values stored in the database. AgentOne calls Hyperic SIGAR API to generate system information, and then it calls the local information manager. Then it reads the node list to find out remote hosts addresses. AgentOne creates AgentTwo and sends AgentTwo to these addresses. AgentTwo needs the address of server node as it sends back AgentThree from remote nodes. So AgentOne sends the URL of server to the AgentTwo. If sending of a AgentTwo to an address fails, AgentOne disposes that AgentTwo. That address is declared unreachable and updated to database. These processes go on in a loop until user stops it by disposing AgentOne.

b) Design of AgentTwo

Figure 5 shows the workflow of AgentTwo:

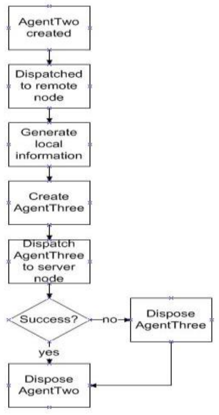

AgentTwo created
Dispatched to remote node
Generate local information
Create AgentThree
Dispatch AgentThree to server node
Success?
no
Dispose AgentThree
yes
Dispose AgentTwo

Figure 5 Workflow of AgentTwo

Though AgentTwo starts its life in server node its original working is done in remote node. On remote node AgentTwo calls the Hyperic SIGAR API to generate resource information. Then it creates AgentThree and sends it to server node. If dispatching of AgentThree to server fails, AgentTwo disposes AgentThree. At last AgentTwo disposes itself.

c) Design of AgentThree

Figure 6 shows the workflow of AgentThree:

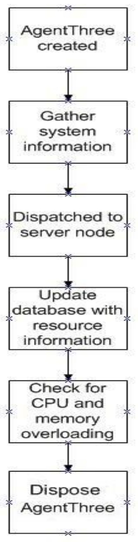

AgentThree created
Gather system information
Dispatched to server node
Update database with resource information
Check for CPU and memory overloading
Dispose AgentThree

Figure 6 Workflow of AgentThree

AgentThree is created by AgentTwo on remote node. AgentThree reads the files created by Hyperic SIGAR and initialize its variables by the values. As aglets supports weak mobility, these data remains intact after AgentThree is dispatched to server node by AgentTwo. On server node AgentThree updates the database. It also checks if any remote host has idle CPU and used memory size below threshold value. The threshold value is defined in the AgentThree. If overloading found then an error is updated in database. At last AgentThree disposes itself.

3.2 Deployment Details

The monitoring system is deployed in the Research Lab of Computer Science and Engineering Department of Thapar University. Five systems are used in deployment. The operating system of these systems is Fedora Core 8. Globus is installed in all those system. To deploy the monitoring system aglets and Hyperic SIGAR are needed to be installed in those systems. These two are installed in the /opt folder. The shell script files to generate resource information using Hyperic SIGAR are stored in a folder named Monitor. The folder is copied to /opt/aglets/cnf. This folder also contains the Java file to process local system information i.e. local Information Manager. The agent codes are copied to /opt/aglets/public folder. All these codes are needed to be compiled. In the server node, /opt/aglets/cnf folder contains a file named ip.txt which contains the addresses of the nodes. The owner of this ip.txt file is needed to be changed to apache. The following sub-section presents a discussion about the creation and deployment of the mobile agents.

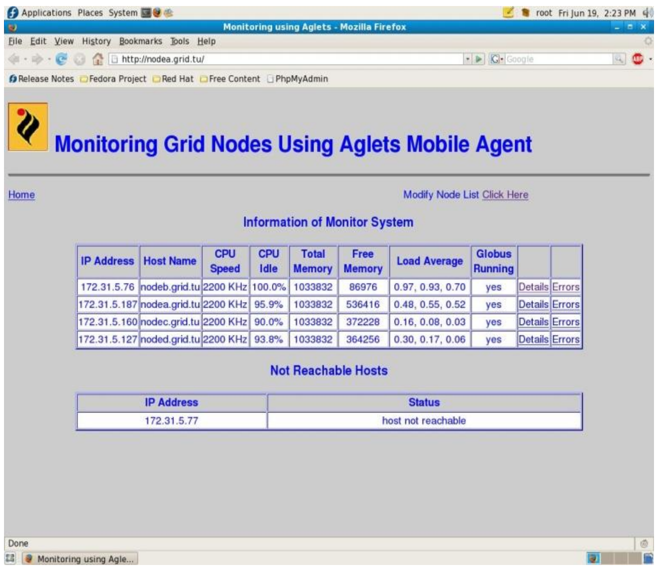
The monitoring system starts its execution when user manually fed AgentOne to the Tahiti Server on the server node. It can be said reversely that in which system user initiates the AgentOne, that system becomes the server node of the monitoring system. AgentOne continually run on the server node. The next section discusses about the experimental results of the monitoring system.

3.3 Experimental Results

The monitoring system uses website based approach to provide monitoring information to the user. This section provides details of various webpages and the monitoring data they provide to the user.

Figure 7 shows the main page. It provides general and brief information about the nodes like IP address, hostname, CPU speed, and percentage of idle CPU, total memory, free memory available, load average and status of globus i.e. globus is running or not on the system.

Figure 8 shows the pop up window to maintain list of monitored nodes. IP address of nodes can be added or removed from the list. Changes in this pop up window results in changes in the ip.txt file of /opt/aglets/cnf/Monitor folder. IP of nodea.grid.tu is not showing in the IP list as nodea.grid.tu is the server node. Server node's information is managed by local information manager. Agents need not to be dispatched to this system, so IP list does not contain the server IP address.


Applications Places System
root Fri Jun 19, 2:23 PM
Monitoring using Aglets - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://nodea.grid.tu/
Release Notes Fedora Project Red Hat Free Content PhpMyAdmin

Monitoring Grid Nodes Using Aglets Mobile Agent

Home

Modify Node List Click Here

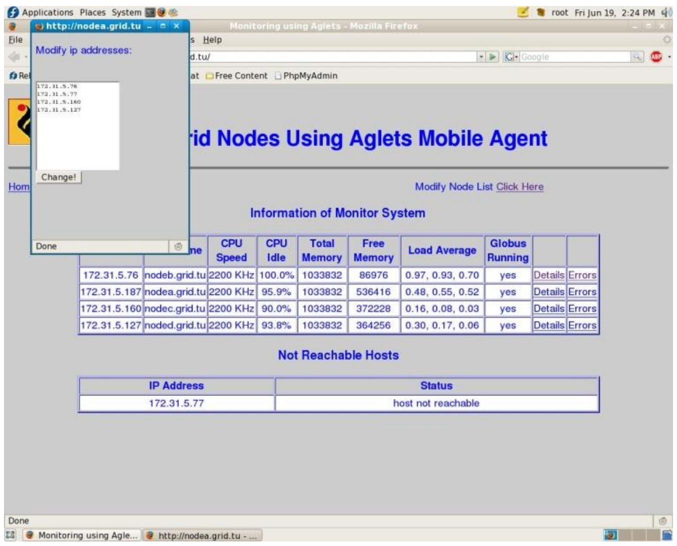
Information of Monitor System

IP Address	Host Name	CPU Speed	CPU Idle	Total Memory	Free Memory	Load Average	Globus Running		
172.31.5.76	nodeb.grid.tu	2200 KHz	100.0%	1033832	86976	0.97, 0.93, 0.70	yes	Details	Errors
172.31.5.187	nodea.grid.tu	2200 KHz	95.9%	1033832	536416	0.48, 0.55, 0.52	yes	Details	Errors
172.31.5.160	nodec.grid.tu	2200 KHz	90.0%	1033832	372228	0.16, 0.08, 0.03	yes	Details	Errors
172.31.5.127	noded.grid.tu	2200 KHz	93.8%	1033832	364256	0.30, 0.17, 0.06	yes	Details	Errors

Not Reachable Hosts

IP Address	Status
172.31.5.77	host not reachable

Figure 7 Main Page



Modify ip addresses:

```
172.31.5.76
172.31.5.77
172.31.5.160
172.31.5.127
```

id Nodes Using Aglets Mobile Agent

Modify Node List Click Here

Information of Monitor System

		CPU Speed	CPU Idle	Total Memory	Free Memory	Load Average	Globus Running		
172.31.5.76	nodeb.grid.tu	2200 KHz	100.0%	1033832	86976	0.97, 0.93, 0.70	yes	Details	Errors
172.31.5.187	nodea.grid.tu	2200 KHz	95.9%	1033832	536416	0.48, 0.55, 0.52	yes	Details	Errors
172.31.5.160	nodec.grid.tu	2200 KHz	90.0%	1033832	372228	0.16, 0.08, 0.03	yes	Details	Errors
172.31.5.127	noded.grid.tu	2200 KHz	93.8%	1033832	364256	0.30, 0.17, 0.06	yes	Details	Errors

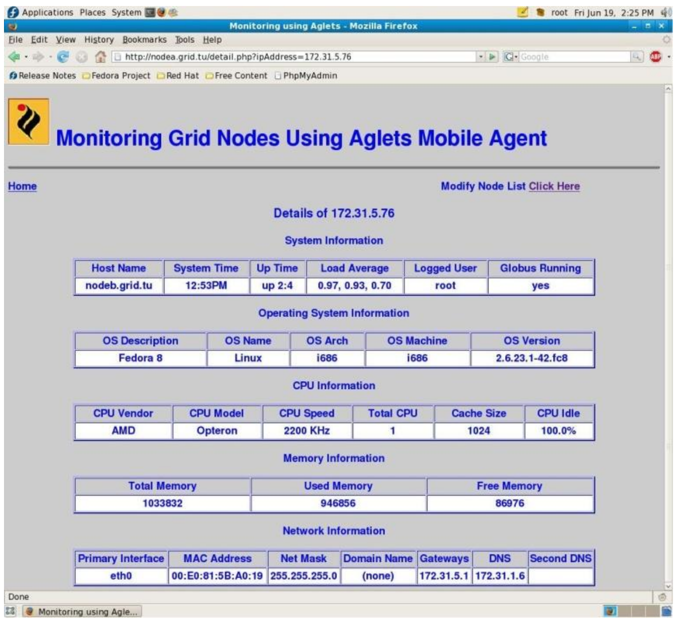
Not Reachable Hosts

IP Address	Status
172.31.5.77	host not reachable

Figure 8 Modifying Node List of Monitoring System

On main page there is two category of information of system. In first category the reachable hosts are listed and information about each reachable host is showed in rows. The row has two hyperlink named Details and Errors. Second category shows the list of unreachable hosts.

Clicking on Details hyperlink on main page will open a page which shows details of a particular node, shown in Figure 9. This page shows details of the node having associated IP address. Details about system, operating system, CPU, memory and network are listed on this page.


Applications Places System
root Fri Jun 19, 2:25 PM
Monitoring using Aglets - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://nodea.grid.tu/detail.php?ipAddress=172.31.5.76
Release Notes Fedora Project Red Hat Free Content PhpMyAdmin

Monitoring Grid Nodes Using Aglets Mobile Agent

Home

Modify Node List Click Here

Details of 172.31.5.76

System Information

Host Name	System Time	Up Time	Load Average	Logged User	Globus Running
nodeb.grid.tu	12:53PM	up 2:4	0.97, 0.93, 0.70	root	yes

Operating System Information

OS Description	OS Name	OS Arch	OS Machine	OS Version
Fedora 8	Linux	i686	i686	2.6.23.1-42.fc8

CPU Information

CPU Vendor	CPU Model	CPU Speed	Total CPU	Cache Size	CPU Idle
AMD	Opteron	2200 KHz	1	1024	100.0%

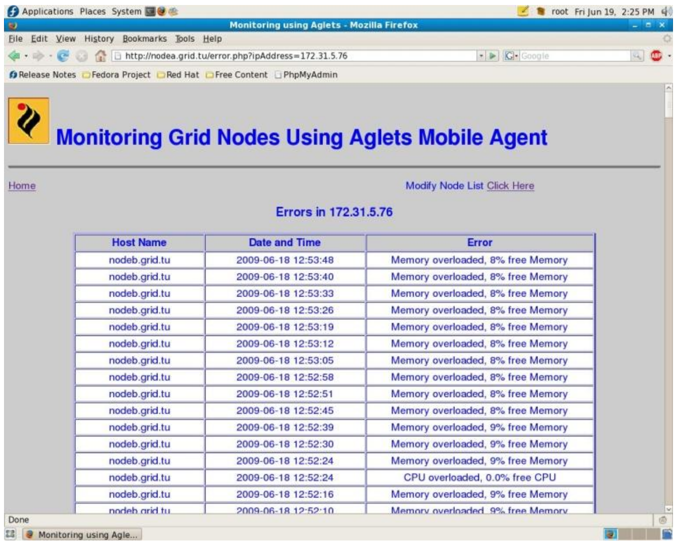
Memory Information

Total Memory	Used Memory	Free Memory
1033832	946856	86976

Network Information

Primary Interface	MAC Address	Net Mask	Domain Name	Gateways	DNS	Second DNS
eth0	00:E0:81:5B:A0:19	255.255.255.0	(none)	172.31.5.1	172.31.1.6	

Figure 9 Detail Information of Node


Applications Places System
root Fri Jun 19, 2:25 PM
Monitoring using Aglets - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://nodea.grid.tu/error.php?ipAddress=172.31.5.76
Google
Release Notes
Fedora Project
Red Hat
Free Content
PhpMyAdmin
Done
Monitoring using Agle...

Monitoring Grid Nodes Using Aglets Mobile Agent

Home

Modify Node List Click Here

Errors in 172.31.5.76

Host Name	Date and Time	Error
nodeb.grid.tu	2009-06-18 12:53:48	Memory overloaded, 8% free Memory
nodeb.grid.tu	2009-06-18 12:53:40	Memory overloaded, 8% free Memory
nodeb.grid.tu	2009-06-18 12:53:33	Memory overloaded, 8% free Memory
nodeb.grid.tu	2009-06-18 12:53:26	Memory overloaded, 8% free Memory
nodeb.grid.tu	2009-06-18 12:53:19	Memory overloaded, 8% free Memory
nodeb.grid.tu	2009-06-18 12:53:12	Memory overloaded, 8% free Memory
nodeb.grid.tu	2009-06-18 12:53:05	Memory overloaded, 8% free Memory
nodeb.grid.tu	2009-06-18 12:52:58	Memory overloaded, 8% free Memory
nodeb.grid.tu	2009-06-18 12:52:51	Memory overloaded, 8% free Memory
nodeb.grid.tu	2009-06-18 12:52:45	Memory overloaded, 8% free Memory
nodeb.grid.tu	2009-06-18 12:52:39	Memory overloaded, 9% free Memory
nodeb.grid.tu	2009-06-18 12:52:30	Memory overloaded, 9% free Memory
nodeb.grid.tu	2009-06-18 12:52:24	Memory overloaded, 9% free Memory
nodeb.grid.tu	2009-06-18 12:52:24	CPU overloaded, 0.0% free CPU
nodeb.grid.tu	2009-06-18 12:52:16	Memory overloaded, 9% free Memory
nodeb.grid.tu	2009-06-18 12:52:10	Memory overloaded, 9% free Memory

Figure 10 Overloading Errors on Node

Figure 10 shows the overloading error occurs in a node. It can be opened by clicking on Errors hyperlink of main page. It shows errors regarding CPU and memory overloading with time and date.

The monitoring system has been tested using the following test cases:

Test Case1 New node is added to the monitoring system.

Test Case2 Node is removed from the node list.

Test Case3 A system is shut down while monitored.

Test Case4 Tahiti server of a node is closed while monitored.

Test Case5 A node is highly overloaded while monitored.

The results of these test cases are:

Test Result1 The system information of the added system is shown as in Figure 7 if the node is up and Tahiti is running. If not then the node is listed under unreachable hosts.

Test Result2 The node is removed from the web pages also. It may take some time as the node's entry from the database is removed by truncating the database.

Test Result3 The node is added to the list of unreachable hosts.

Test Result4 The node is added to the list of unreachable hosts though the node is live and network is also fine. Aglets needs Tahiti server to communicate. As Tahiti is closed, agents from server cannot communicate with this node. As a result this node is declared as unreachable.

Test Result5 As the node is highly overloaded, the value of free memory and free CPU become low. They gone below the threshold value predefined in the AgentThree. As AgentThree finds overloading, table regarding to the error of database is updated. This overloading information can be accessed using the web pages as shown Figure 10.

4 Conclusions

Grid computing suffers from various problems for its dynamicity and supports for various technologies and heterogeneous resources. The grid environment needs to be monitored to discover newly added resources, choosing a system optimum to run a job and detect any failure. A mobile agent based monitoring system for grid computing is proposed and discussed in this paper. The monitoring system is easy to deploy, use and scalable. It provides information to the user which is easy to understand and access. If server node fails, the monitoring system can be recovered by introducing a new server just by feeding AgentOne to the node.

5 References

[1] I. Foster, C. Kesselman, et. al, "The Grid: Blueprint for a Future Computing Infrastructure," Morgan Kauffmann Publishers,Inc, San Francisco, California,Second Edition, 2003.

[2] Grimshaw A., Wulf W. et al. "The Legion Vision of a Worldwide Virtual Computer". Communications of the ACM, Vol 40(1), January 1997.

[3] Object Management Group (OMG), http://www.omg.org/

[4] I. Foster, C. Kesselman, S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," International Journal of Supercomputer Applications, 15(3), pp.115-128, 2001.

[5] P. Townend, J. Xu, "Fault Tolerance within a Grid Environment," In Proceedings of AHM2003, page 272, 2003.

[6] S. Zanikolas, R. Sakellariou, "A Taxonomy of Grid Monitoring Systems", Future Generation Computer Systems, Vol 21(1), pp 163-188, January 2005.

[7] M. Mansouri-Samani, M. Sloman, "Monitoring Distributed Systems", IEEE network 7 (6), pp 20-30, 1993.

[8] R.L. Ribler, J.S. Vetter, H. Simitci, D.A. Reed, "Autopilot: Adaptive Control of Distributed Applications", Proceedings of the Seventh IEEE Symposium on High-Performance Distributed Computing, pp. 172–179, 1998.

[9] A. Hawkeye,"Monitoring and Management Tool for Distributed Systems”, http://www.cs.wisc.edu/condor/hawkeye/.

[10] B. Tierney, D. Gunter, “NetLogger: A Toolkit for Distributed System Performance Tuning and Debugging”, G.S. Goldszmidt, J. Schonwalder (Eds.), Proceedings of the IFIP/IEEE Eighth International Symposium on Integrated Network Management (IM 2003), Vol 246 of IFIP Conference Proceedings, Kluwer, pp 97–100, 2003.

[11] I. Foster, “Globus Toolkit Version 4: Software for Service-Oriented Systems”, IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779, pp 2-13, 2006.

[12] Hyperic SIGAR, http://www.hyperic.com/products/sigar.html.

[13] D. B. Lange, M. Oshima, “Mobile Agents with Java: The Aglet API”, World Wide Web, Vol 1(3), pp 111 – 121, 1998.

[14] D. B. Lange, “Mobile Objects and Mobile Agents: The Future of Distributed Computing?” Lecture Notes in Computer Science, ECOOP’98-Object-Oriented Programming, 1998.

Workstations as Grid with enabled "Green Provisioning" in a PBS Professional embedded HPC Cluster

Jayant Mukherjee[1] **and Avadhesh Mittal**[2]
[1]Project Manager, Altair Engineering, Bangalore, Karnataka, India
[2]Sales Manager, Altair Engineering, Bangalore, Karnataka, India

Abstract – *The paper provides the insight of algorithm build on a HPC cluster where PBS is deployed. Paper states on the power of using workstations as node when required for the grid environment and switch off if not in use. This helps uninterrupted use of workstation in a need basis else switch off the machine provided there is no interaction by the end user. Paper also describes the cost factor involves in maintaing HPC and workstations and how using the proposed algorithm above PBS can harness the power consumption and grid farms usage. Message is "Go Green"*

Keywords: Green Provisioning, PBS, Auto Restart, OS Provisioning, ReQueuing

1 Introduction

With the introduction of quad core and soon to arrive multi-core workstations(ws) organizations have a challenge and a fantastic opportunity to harness this extra processing power.

Workstation Grid, an algorithm embedded on HPC Cluster managed by Altair's PBS GridWorks Product suit allow organizations to utilize the extra cores to gain a real competitive advantage, whether its getting products to market quicker, or running more analysis cycles in the same timeframe etc.

PBS which is a backend Grid Engine provides the compute power to carry out complex jobs and handles organization's return on hardware investment for an HPC setup. Typical office productivity tools such as email, web browsing, spreadsheets and presentations do not require multi-core processing power and in many cases are not able to use it anyway. Multiply this across many users and departments and its clear there is potentially a very large pool of untapped processing power.

Workstation Grid harness all the processing power across workstations, and provide a controlled and managed system to allow jobs to use spare CPU cycles without affecting any work being carried by users.

1.1.1 HPC Environment

HPC setup could be of two types:

- Dedicated high performance computing (HPC) resources such as clusters, large shared memory systems etc
- Running computations on desktop workstations

Altair's PBS GridWorks has the capacity to manage both the types with required scheduling policies and management tools to maximize job through put, maximize resource utilization and minimize job execution time

2 Current Problems

In the world of new jargons like SaaS, PaaS, IaaS, Cloud Computing & Virtualization, the problem still persists is how to handle the operating cost involved in maintaining the workstations which consume huge power and cooling cost. Also in case of an HPC set up if the node has to be increased from X to nX, the corresponding maintenance through cooling, electricity (power) also increases tremendously.

Though these jargons reveal an illusion of how to compensate through different services, the bottom line is "How effectively an organization can utilize the underlying Workstations?" This issue gets multiplied with organizations having workstations which has better performance oriented infrastructures plus an HPC setup. This is a typical problem occurred in Product Design & Development group of Altair India where 2000 cores of HPC setup sitting on PBS professional for job submission and monitoring of different CAE jobs includes around 240 workstations each with high performance oriented architecture.

The below figures show how the HPC setup gets unutilized (white HPC nodes are always active even if no jobs are running)

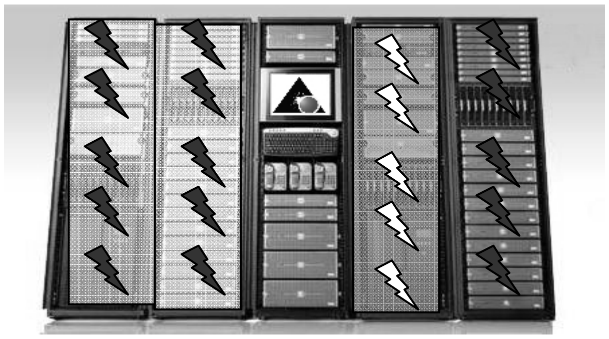


Figure 1: Typical Organization HPC Setup (Black Nodes in use, white nodes are unused)

2.1 How much electricity do computer use?

A typical desktop computer uses about 65 to 250 watts. A computer whose label or power supply says 300 watts might only use about 70 watts when it's actually running, and only 100 even in peak times with serious number-crunching and all the drives spinning.
As long as the computer goes into sleep/standby when it is not in used, the computer doesn't use squat for electricity.
Of course, one should absolutely make sure that the computer is set to sleep automatically when user is not using it. This requires the user discipline and organization policy.

Table 1: Watts Consumption of different computers

Computers	
Desktop Computer	60-250 watts
On screen saver	60-250 watts (no difference)
Sleep / standby	6 watts
Laptop	15-45 watts
Monitors	
Typical 17" CRT	80 watts
Typical 17" LCD	35 watts
Screen saver (any image on screen)	same as above (no difference)
Sleeping monitor (dark screen)	up to 15 watts

2.1.1 Cost Estimation for running a computer

To calculate our costs use this formula:

$$\frac{\text{Watts x Hours Used}}{1000} \text{ x Cost per kilowatt-hour = Total Cost}$$

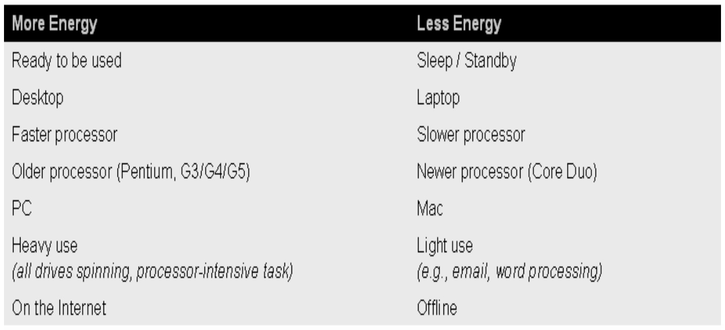
2.1.2 Worst Scene

WorkStation Case (Calculation based on currency INR) : A big high-end computer with a high end graphics card and an old CRT monitor, and we leave them on 24/7 :

330 watts x 24 hours x 365 days/yr = 2,890,800 watt-hours, or 2891 kilowatt-hours.
If we're paying ~ INR 5 per kWh, total payment is ~ INR 14500 a year to run our computer

Practical Case (Calculation based on currency INR): A normal PC with office application use, which uses about 105 watts, and we are smart enough to turn it off when we're not using it. We use it for 10 hours a day, five days a week. That's 50 hours a week, or ~ 2600 hours a year. So our 105 watts times 2600 hours = 273 000 watt-hours. Divide by 1000 and we have 273 kilowatt-hours (kWh).

If we're paying ~ INR 5 per kilowatt-hour, toal payment is ~ INR 1365 a year to run our computer
But : - The practical does not happen so we remain in between on the spent
Factors affecting Energy Use:



More Energy	Less Energy
Ready to be used	Sleep / Standby
Desktop	Laptop
Faster processor	Slower processor
Older processor (Pentium, G3/G4/G5)	Newer processor (Core Duo)
PC	Mac
Heavy use *(all drives spinning, processor-intensive task)*	Light use *(e.g., email, word processing)*
On the Internet	Offline

We need to add another 35 watts for an LCD monitor, or 80 watts if we have an old-school CRT. And we should not forget about the related Devices also.

*Laptop computers use about 15-45 watts, far less than desktops.
Sleep & ScreenSavers- Are they Best Solution?
When our computer sleeps (aka "standby", "hibernate") the computer uses up to 10 watts. (So does the monitor.)
We can set our computer to sleep automatically after a certain amount of idle time. Setting our computer to auto-sleep is the best and easiest way to save on computer energy use!
A screensaver that shows any image on the screen doesn't save any energy at all -- we save energy only if the monitor goes dark by going to sleep

2.2 Sample Calculation

Usage hours – 10
Computer - 125 Watt/Hour + Monitor - 100 watt/hour
Non – Usage hour 14
Sleep/standby – computer - 10 watt/hour+ Monitor – 10 Watt/hour
Screensaver –
Computer – 125 Watt/hour + Monitor – 10 Watt/hour
About 250 Working day/year
10*250 ~= 2500 Hour/usage
Total Hours in a year
24*365 = 8760 Hours
About 6000 Hours of non-usage (→wastage of power)
Sleep/standby enabled power – 6000X20 = 120000 Watts
Screensaver enabled power – 6000X135 = 810000 watts
Switching off for about 4500 Hours with Green computing –
Sleep enabled comp = 4500 X 25 = 112500 Watts/PC/Year
Screen Saver Comp = 4500 X 135 = 607500 Watts/PC/Year

3 Benefits of WS Grid Environment

This paper demonstrates that a HPC cluster where PBS is the Grid Engine, Workstation Grid is able to harness the increasing power of spare cores and spare CPU cycles on workstations through workstation clusters will have a massive increase in compute capability with no additional hardware cost.

Even with dedicated HPC systems in place the benefits of workstation Grid are achieved by

- Offloading small (1 or 2 CPU jobs) from the dedicated system
- Offloading short running jobs from the dedicated system
- Offloading less critical jobs from the dedicated system
- Offloading test jobs to check a solution is going to converge from the dedicated system

Dedicated HPC systems are expensive resources that should be made to work as hard as possible on the most important jobs for the business. If less critical or smaller workloads can be offloaded to workstation clusters thus improving HPC utilization and increasing overall job throughput then that organization will get a much better return on their IT investment and potentially gaining competitive advantage. Harnessing the power of desktop systems using workstation clusters brings many benefits and compliments the dedicated HPC resources with no additional hardware costs.

3.1 High Performance Computing: GOALS

- o Optimal utilization of hardware and solvers
- o Shorter time to market:-the same set of computation to take less time.
- o Higher product quality :-achieve more complex computation in the same time

3.2 Management Information

- o Know the overall utilization of the hardware and the software solvers
- o Enable well founded planning for future purchase of any hardware or solver license extension

3.3 IS Management

- o A self sufficient system to reduce day to day administration of HPC
- o HPC User's
- o Easy access to the compute resources

4 Features of PBS managed WS Grid

A workstation Grid is a collection of workstations that are networked together and uses a workload manager PBS/LSF/SunGE/Moab/Torque to control & monitor the resources within the cluster, and to manage the deployment or scheduling of jobs across the workstations.

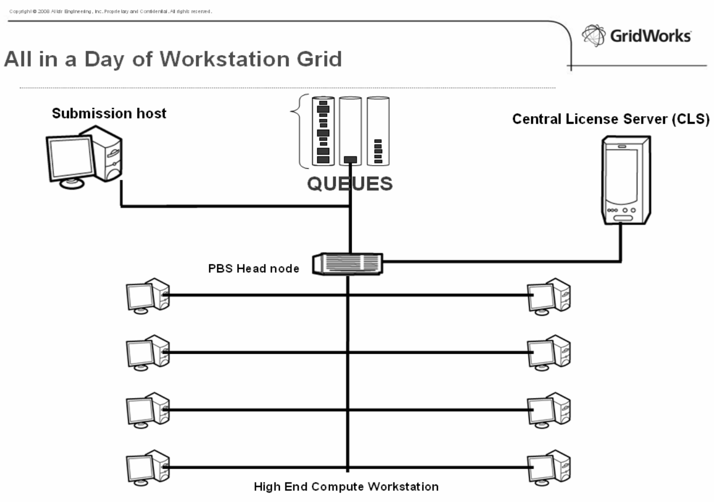

All in a Day of Workstation Grid
GridWorks
Submission host
QUEUES
Central License Server (CLS)
PBS Head node
High End Compute Workstation

Figure 2: A typical Workstation Grid Setup

4.1.1 Auto Detection of Nodes in WS Grid

Altair's PBS server does deployment of jobs and to recover in the event of workstation or network failures a algorithm is written on PBS which provides a single point of entry into the workstation cluster through PBS. Jobs submitted to the PBS are held in queues. Algorithm which sits on the PBS server continually monitors resource availability around the cluster e.g. available CPUs/cores, memory etc and compares this to resources required by queued jobs. As soon as a workstation is found in a logged off state it becomes available for a job is dispatched to run on a workstation in the cluster.

Desktop Grid submits jobs to :

- ➢ A logged off workstation only
- ➢ Auto detection system finds out which workstation is in logged off condition

Ensuring

- ➢ Users are not affected by background jobs

Possibility

- ➢ To use multi core cpu power for batch jobs at the same time reserve some for interactive jobs

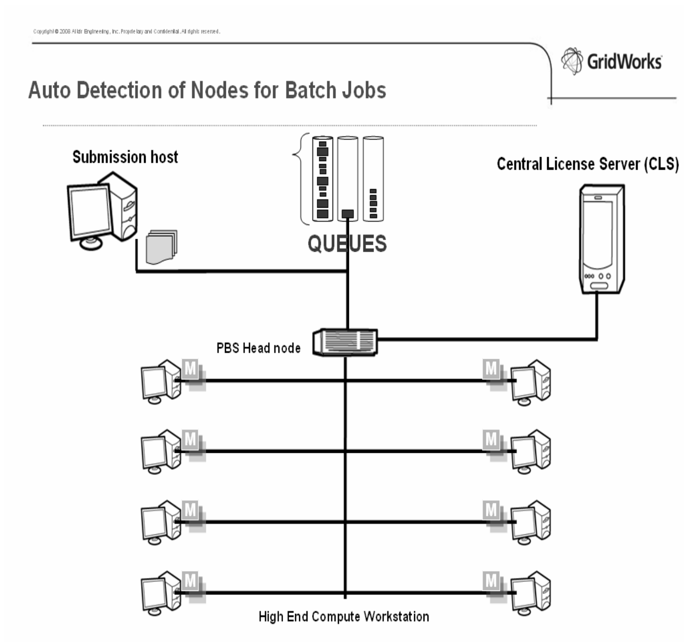

GridWorks
Auto Detection of Nodes for Batch Jobs
Submission host
QUEUES
Central License Server (CLS)
PBS Head node
M
High End Compute Workstation

Figure 3 : Workstations acting as nodes in a HPC environment

PBS provides the features for all internal communication within the cluster to monitor the status of all workstations and the progress of running jobs.

It is essential that a user's experience is not changed when their workstation is configured as part of a Grid. However, spare cycles when the user is not using their system become available for the workstation cluster to run other jobs. Once the user leaves their system and their activities have finished the workstation is flagged as free and becomes an available node in the cluster.

4.1.2 Automatically makes WS offline

Algorithm written on PBS Pro marks a workstation offline

- ➢ When a user logs on to a workstation
- ➢ auto-detection system finds out which workstation is in logged on state

Ensuring

- ➢ PBS server do not submit jobs on a workstation which is used by a user

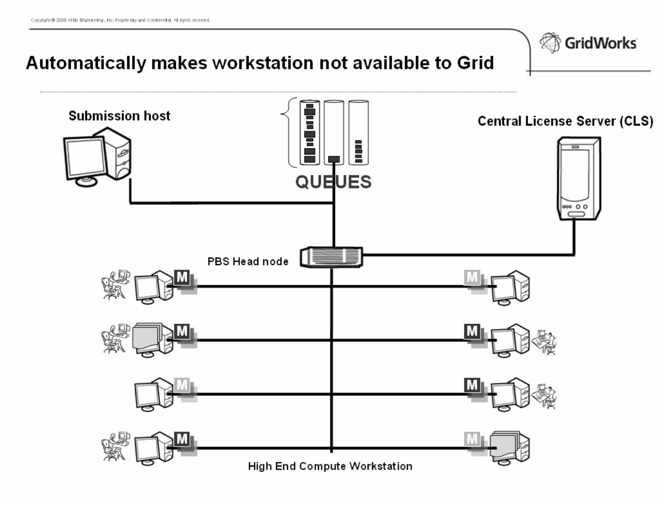

GridWorks
Automatically makes workstation not available to Grid
Submission host
QUEUES
Central License Server (CLS)
PBS Head node
M
High End Compute Workstation

Figure 4 : Auto ShutDown of Nodes not in use by end user & by master node (3 nodes left, 2 nodes right)

4.1.3 Auto Migration/requeue jobs to an available WS

If cluster jobs are still running on a workstation when the user returns, algorithm gives direction to PBS to make a decision what to do with the jobs.

- Migrate/requeue running jobs
- Suspend running jobs until the workstation becomes free again. Then resume the suspended jobs so they continue to execute
- Let the running jobs continue to run

4.1.4 Auto log off a ideal WS to make it available to Grid:

An interesting difference between workstation clusters and typical dedicated HPC systems is that the nodes in a workstation cluster monitor user interactivity and workload and report this back to the PBS as available. Dedicated HPC systems have nodes that are not used interactively so don't need to monitor what local users are doing.

Auto logoff feature

Will automatically log off a workstation with

1. No key board event
2. No cpu activity
3. No local jobs on the workstation for Time 'T'

Ensuring

- ➢ The unutilized workstation becomes part of Desktop Batch Grid

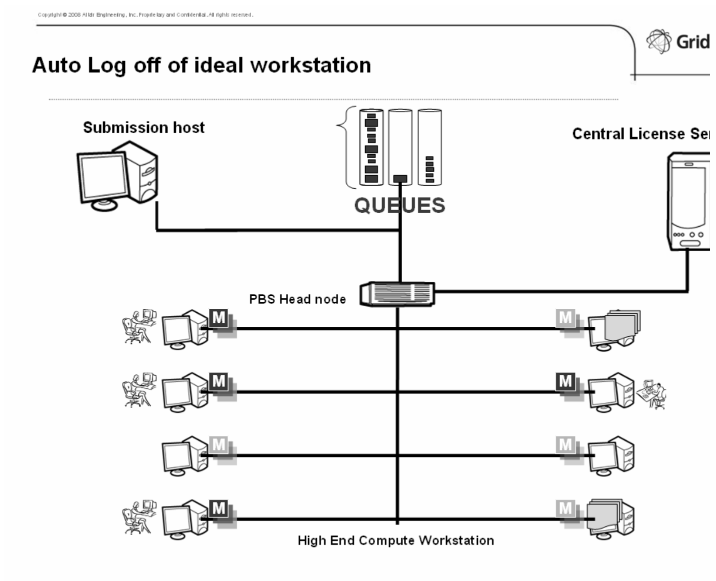

Auto Log off of ideal workstation
Grid
Submission host
QUEUES
Central License Se
PBS Head node
M
High End Compute Workstation

Figure 5 : Auto Log off of ideal workstation and making available for Grid (Node 3 gets active from right)

4.1.5 Green Computing – Auto Shutdown/Reboot as needed:

With increasing energy costs and more attention to "green issues" many organizations have policies to turn off unused workstations overnight. PBS Pro provides power aware scheduling to turn off system when not in use and at the same time use WOL feature to boot required no. of workstations for the jobs in queue.
Green Computing will automatically shutdown a workstation if

1. Workstation is logged off
2. No batch jobs are running for Time 'T' (Configurable)

Power aware scheduling can boot workstation as and when needed as per the jobs resource requirement
Ensuring

- ➢ The unutilized workstation either locally or from batch system is shutdown – no user intervention

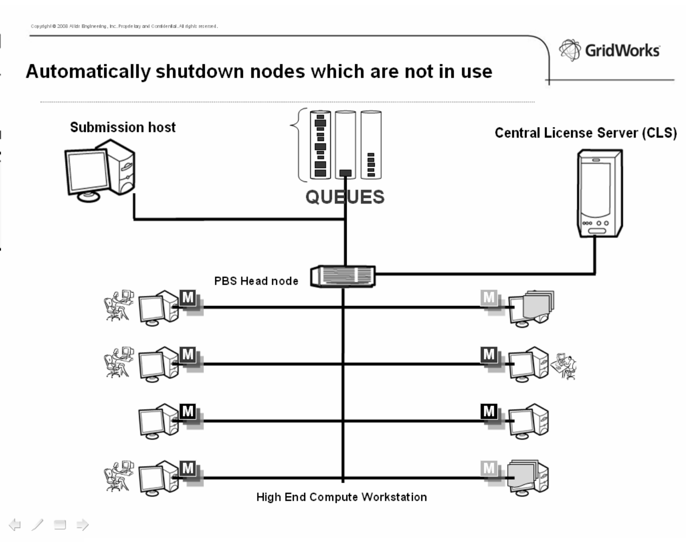

Automatically shutdown nodes which are not in use
GridWorks
Submission host
QUEUES
Central License Server (CLS)
PBS Head node
M
High End Compute Workstation

Figure 6: Shutdown of nodes if not used by user & HPC (Number 3 node of both sides)

4.1.6 OS Provisioning to use WSs either in Windows/Linux mode for Batch Jobs:

The script (algorithm) written on PBS Pro uses Power aware scheduling which remotely sends signal for Workstation boot and shutdown if not in use.
To gain advantage of running a Job in Linux or Windows mode
The workstations could be made dual boot

- ➢ The default boot is in Linux to cluster the workstation.
- ➢ Auto shutdown feature in the algorithm will direct PBS to shutdown the workstation and it will be booted in Linux as and when required.
- ➢ MPI jobs can run using PBS Grid Setup
- ➢ Once the job in Linux is over, the Auto shutdown will shut down the workstation
- ➢ Users can boot the workstations in Windows mode for interactive application

Ensuring

- No intervention from user or System admin team is required for switching workstation to Linux
- The unutilized workstations either locally or from batch system are shutdown – no user intervention

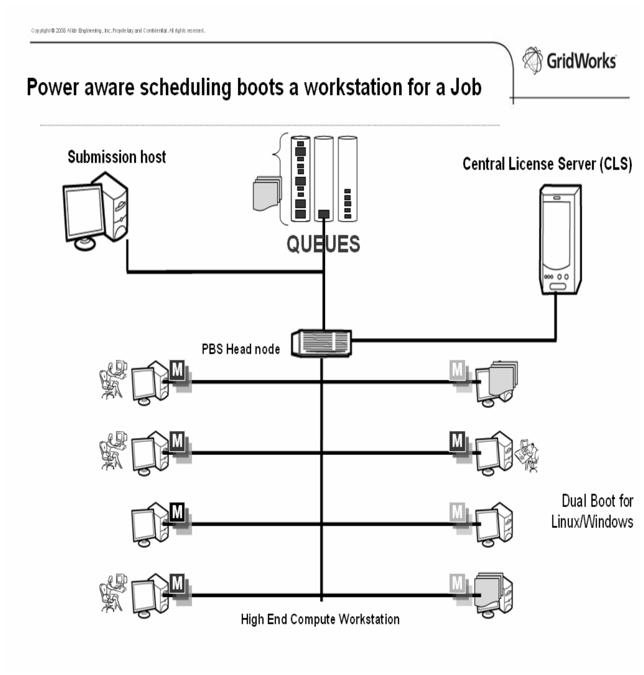

Copyright © 2008 Altair Engineering, Inc. Proprietary and Confidential. All rights reserved.
GridWorks
Power aware scheduling boots a workstation for a Job
Submission host
QUEUES
Central License Server (CLS)
PBS Head node
M
Dual Boot for Linux/Windows
High End Compute Workstation

Figure 7: Dual boot mechanism: Linux as Node, Windows as End user usage

5 Conclusions & Further Work

In this paper, a solution has been given on how to effectively use organization workstations and make the power of workstations as part of HPC rather than planning and procuring for new HPC setup. The paper also describes that using algorithm as part of PBS setup can make not only huge power save but can utilized unused resources and can shut down if not in use.

The complete setup is ready and running at Altair India, PDD group where the CAE jobs are required to be submitted in an existing HPC setup. When HPC is fully used and there are available workstations, algorithm switch on the idle workstations and use it for calculations. Simultaneiously it also checks for idle workstations and hpc nodes and shut it down. If CAE analysts starts the work on his workstation, algorithm in the backend switch over to his preferred OS and migrate the job to next available grid node fulfilling the criteria of submission or makes it available for the resources to get free.

Future work involves complex environment where organization have multiple grid engines following hpc profiling. Using the logic of grid engines compatibility, it is possible to use the same algorithm for non-PBS environment where PBS becomes master Grid Engine driving all other engines through hpcprofiling.

6 References

[1] Sites & Support documents of PBS GridWorks by Altair Engineering

Feedback guided job modeling in PRAGMA environment

Madhulina Sarkar[2], Rupam Mukhopadhyay[1], Dibyajyoti Ghosh[1], Sarbani Roy[1], Nandini Mukherjee[1]
[1]Department of Computer Science and Engineering, Jadavpur University, Kolkata – 32,
[2]Department of Computer Sc. and Engg., Govt. College of Engineering and Leather Technology, Kolkata - 98,
madhulina.sarkar@gmail.com, rupam.mukhopadhyay@gmail.com, dibyajyotig@gmail.com,
sarbani.roy@cse.jdvu.ac.in, nmukherjee@cse.jdvu.ac.in

Abstract: *Grid computing environment provides access to more computational resources than ever before by aggregating and utilizing the computational power of ensembles of shared, heterogeneous, and distributed resources for executing computation intensive jobs***.** *Moreover, Grid provides a highly dynamic environment in which resources/services can be added or withdrawn at any point of time. Managing such dynamic, heterogeneous pool of resources and allocating the resources to the jobs in most effective manner is one of the central concerns in Grid environment. Thus, resource management is one of the focus areas in Grid computing research. One major objective of resource management in a computational Grid environment is to allocate jobs to various available resources according to jobs' requirements to make efficient use of the computational resources under different resource providers and, thereby, achieve high performance of the jobs. However, heterogeneity of resources and jobs makes it difficult to manage the execution of jobs in such environments. A proper job modeling can be helpful to allocate jobs into their most suitable resource providers in Grid. Consequently, it assists to manage the execution of jobs as well as helps to achieve the high performance. This paper presents a feedback guided job modeling technique that describes the process required to identify the most suitable resource provider for a particular job.*

Keywords: Grid, resource management, multi-agent system, job modeling

1. Introduction

Modeling resource requirements for a job has been a major challenge for the researchers for a long time. The runtime behavior of a job is generally not known beforehand. Furthermore, complexity of the underlying platform and changing resource status exacerbate the problem of estimation of resource requirements for jobs.

In spite of the complexity, particularly modeling the resource requirements for every job is important in a large distributed computational environment, where heterogeneous hosts are available. Precise modeling enables the client to schedule the job on a suitable host and thereby maximize the performance of the job. In recent times, Grid has emerged as large distributed environment with characteristics which are very different from traditional distributed systems [14]. A Grid environment comprises heterogeneous resources administered by multiple administrative domains. When a client submits a job in such environment, the client must specify resource requirements, so that a resource broker can find appropriate resources for carrying out the job and achieve the performance guarantee as desired by the client. In most resource management systems for Grid [15], the client is left to decide the resource requirements for their jobs. Such ad hoc decisions taken by the client may often lead to overestimation or underestimation of the resource requirements.

In our previous works [18], we proposed a framework for adaptive execution of jobs in Grid environment. Within the framework, an application or the components of an application (referred as jobs) are initially scheduled onto suitable resource providers. Later, on the basis of the performance properties, the jobs are either locally tuned or rescheduled to a different resource provider so that performance guarantee is maintained [18]. At the time of scheduling and rescheduling, estimation of resource requirements is necessary, and we believe that the estimation should be done automatically instead of leaving it to the knowledge and expertise of the client. But as mentioned before, automatic modeling of resource requirements is a difficult task and precise modeling even for a traditional architecture is not feasible. In case of Grid, this becomes further difficult because Grid provides a dynamic environment where resource providers may join or leave any time.

This paper proposes an initial framework for automatic estimation of resource requirements at the time of initial scheduling of a job. The framework is based on the static analysis of the job and the *execution history* of the same job or a similar job which was executed before.

The rest of the paper is organized as follows. Section 2 discusses about job modeling in Grid Environment. Section 3 describes the PRAGMA (Performance based Resource Brokering and Adaptive execution in Grid) environment where we have introduced our feedback guided job modeling module. Section 4 explains the concepts of feedback guided job modeling as has been introduced in PRAGMA. Section 5 compares the effect of feedback guided job modeling in job execution with existing job modeling in PRAGMA environment. Section 6 presents the conclusion and directions for future work.

2. Job Modeling in Grid

General purpose of job modeling is to document the requirements of a job and it forms the basis for later improvement. It is an important phase in Grid environment and it determines the types of resources for a particular job according to its requirements. Thus, job modeling is the initial stage of resource brokering. The data model, used to characterize the requirements of a job is the job model. Proper analysis of the job is required for creating an effective job model.

Majority of the middlewares developed for Grid [[1], [2], [15], [16]] demand that a job is submitted with a specification of its requirements which is explicitly provided by the Grid users. This requirement specification of a job is matched with the available resources in the Grid. If the required resources are found, the job is allocated to the resource provider for its successful execution. Here, clients are allowed to characterize the requirements of a job and a data model containing relevant values (A Job Requirement List (JRL) in [[1], [2], [17]]) is constructed with the help of client provided information. The data model is prepared from the description of each job (Jdesc in [[1], [2], [17]]) and identifies the minimal resource requirements to run that job for meeting the desired QoS. However, two problems can be envisaged here; these are over-provisioning and violation of service level agreement (SLA) that might have been set up between the user and the resource provider at the time of job submission. Over-provisioning is a persistent problem which causes waste of computing resources. On the other hand, a job can not complete its execution successfully on a particular resource provider if service level agreement is violated. Thus, job modeling is an important issue in Grid computing for efficient resource management and for successful execution of jobs in Grid environment.

2.1. Related work

A large number of research works focus on job analysis and on job modeling to predict job performance or to estimate resource requirements on a single parallel supercomputer, cluster or in Grid environment. In [8], Jirada et al proposed a resource estimation model for parallel applications executing in a homogeneous computational cluster environment and developed a parallel-pipeline model of execution that performs a parallel associative operation over a large set of distributed processors [11]. Vikram Sadand Adve [9] analyzed the behavior, performance of parallel programs and developed a simple deterministic model to predict parallel program performance based on influence of random delays in execution time of a job in parallel computers. In [3], Konstantinos Christodoulopoulos et al analyzed the inter-arrival times of jobs and the workload at LCG/EGEE [10] cluster. They proposed a simple model for job arrival processes at the cluster and Grid level. In [5], Li et al used the LCG real time monitor tool [6] to collect data from resource brokers (Rbs), located at different machines at CERN, Germany and UK and proposed traffic models for the job arrival processes. A methodology to analyze and synthesize pseudo-periodic job arrival processes has been proposed in [4]. The methodologies for job modeling discussed in [3], [4], [5] are based on the job arrival processes and the execution times of the jobs. Predicting resource requirements of a job in Grid Computing Environment has been considered in [12] and [13]. In [12], Arshad Ali et al proposed a prediction engine that provides estimates of the resources required by a submitted job on the basis of historical information. This history based approach operates on the principle of identifying similar applications and predicting runtime of similar applications by computing a statistical estimate. They have argued that two applications are similar because the same user on the same machine submitted them or because they have the same application name and are required to operate on same-sized data. In [13], Bohlouli et al proposed a Grid History Based Prediction Architecture which is based on the principle that the architecture itself predicts resource requirements. The job resource requirement prediction algorithm operates on identifying similar jobs based on some similar characteristics of jobs and then a statistical prediction is done in centralized and decentralized way to estimate their execution times.

In our proposal for feedback guided job modeling we mainly concentrate on inherent characteristics of the parallel jobs and their execution environments which are stored in a database as *execution history* for each job which has been executed earlier. From the *execution history*, we attempt to predict a suitable job execution environment and estimated execution time for a newly submitted job.

3. The PRAGMA environment

S. Roy et al [[1], [17]] proposed an integrated framework for performance-based resource management in computational Grid environment. The framework is supported by a multi-agent system (MAS) that has been developed using a firm software engineering approach based on Gaia methodology. The MAS initially allocates the jobs onto different resource providers based on a resource selection algorithm. Later, during runtime, if performance of any job degrades or quality of service cannot be maintained for some reason (resource failure or overloading), the MAS assists the job to adapt to the changed environment. The MAS provides adaptive execution facility either by rescheduling the jobs onto different resource providers or by tuning certain portions of the job locally. Based on the MAS, a tool PRAGMA (Performance based Resource Brokering and Adaptive execution in Grid) [[1], [17]] has been developed.

3.1. Agents in PRAGMA

Altogether six types of agents function in PRAGMA [[1], [17]]. A Broker Agent (BA) is responsible for finding suitable resources for jobs which are submitted to the system by a client. A ResourceProvider Agent (RPA) on a particular resource provider prepares a ResourceSpecificationMemo which stores the details of the available resources. Later, ResourceSpecificationMemos are used by the Broker Agent for resource brokering. A JobController Agent (JCA) (with the help of Broker Agent and ResourceProvider Agent) performs initial resource selection for all jobs on the basis of a resource selection algorithm [2] which ensures that every job gets allocated to a resource provider that can fulfill its requirements and deliver the desired quality of service(QoS). JobController Agent establishes service level agreements (SLA) between the client and the selected resource owners. A set of JobExecutionManager Agents (JEM), one per job in a batch, become associated with the jobs and control and keep track of their execution. JobExecutionManager Agent moves to the resource provider along with the job and when the job executes, it liaises with the Analysis Agents. A number of Analysis Agents are organized in a hierarchy and are employed to carry out performance analysis at various levels of the environment. Analysis Agents are further subdivided as: (1) Grid Agent (GA), (2) Grid Site Agent (GSA), (3) Resource Agent (RA) and (4) Node Agent (NA). An overview of the hierarchical Analysis agent framework is presented in [18]. If the Analysis Agents observe that the performance of a job degrades on a particular resource provider, they either collaborate with JobExecutionManager Agent for migration of the job to a different resource provider or instruct a local Tuning Agent (TA) to take appropriate action for improving the performance of the job [[1], [18]].

3.2. Job modeling in PRAGMA

In the current implementation of PRAGMA, job modeling is not automatic. In this phase, the users are allowed to characterize the requirements of a job. For doing this, the users must have a very good knowledge about the problem domain and the implementation. Also, as the phase is not supported by any data, the characterization cannot be flawless and efficient. This paper proposes a technique to automate job modeling by managing different types of data including the execution-history of every job. Following is a brief description of the proposed methodology for feedback guided automatic job modeling.

Within the PRAGMA environment, the agents described in the previous section are engaged in job modeling phase as well. Interactions of various agents during the job modeling phase are depicted in Figure1 and are explained in Section 4.

4. Feedback guided job modeling

In earlier research works, job modeling is generally based on analysis of various characteristics of jobs and parameters related to the underlying hardware platforms [[4], [5], [8], [9], [12], [13]]. But such information is not always sufficient for developing a suitable model of a job.

In a computational environment, a compute-intensive job is generally executed several times or often it can be observed that a submitted job is similar to some jobs executed earlier. Based on these observations, we propose that job modeling in a computational Grid environment should be feedback guided. During every execution of a job, its *execution history* is stored. This history includes any performance problem observed during the previous execution and the actions taken to overcome the performance problem. This section explains the proposed methodology and the data models used for this purpose.

4.1. Agent interaction

All six agents in multi-agent system (MAS) [[1], [17]] are involved in feedback guided automatic job modeling. Figure 1 depicts collaboration diagram of multiple agents for feedback guided job modeling. Initial job-submission is handled by the Broker Agent. After submission of a job, the Broker Agent analyzes the job before its execution and retrieves static analysis data. Then it requests the Grid Site Agent (GSA) to get *execution history* for a similar type of job, if there is any. The Broker Agent also prepares a *Job Requirement List* (JRL) [[1], [2], [17]] based on static analysis of the job and on historical data supplied by the GSA. Then it prepares Resource Provider List by matching JRL with Resource Specification table which is sent by Resource Provider Agent. Job Controller Agent prepares service level agreement for the job. Job Execution Manager controls the job execution and informs Node Agent about job execution strategy. The Node Agent analyzes the job execution strategy and for any performance problem it contacts to Tuning Agent. Tuning Agent sends tuning action to Job Execution Manager and Job Execution Manager accordingly tunes the job or migrates the job to different resource providers. Again, constructing the database of *execution history* is an important part of feedback guided automatic job modeling. So, all tuning actions that are taken for a job by the JEM and all runtime analysis data for each job are sent to the JCA. JCA forwards the job execution details of each job to the GSA and the GSA stores it as *execution history* database for future use in feedback guided automatic job modeling.

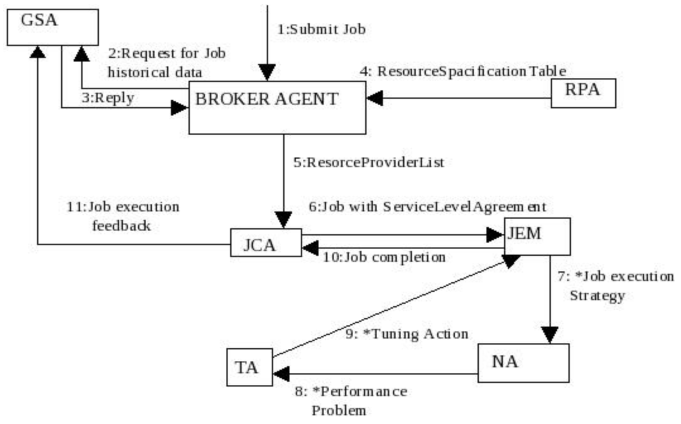

GSA
1:Submit Job
2:Request for Job historical data
3:Reply
BROKER AGENT
4: ResourceSpacificationTable
RPA
5:ResorceProviderList
11:Job execution feedback
6:Job with ServiceLevelAgreement
JCA
JEM
10:Job completion
7: *Job execution Strategy
9: *Tuning Action
TA
NA
8: *Performance Problem

Figure 1: Collaboration diagram of multiple agents for feedback guided job modeling.

4.2. Job models

To model a job, we consider two sets of data – (i) *BE (Before Execution)* set of data that can be retrieved before executing the job and (ii) *AE (After Execution)* set of data that are retrieved after executing (AE) the job. Since, the BE type data set is not machine dependent, they can be categorized as static information of a job, whereas the AE type of data is machine dependent. Figure 2 depicts the class diagram for our proposed data model. The base classes in the data model include Job_Desc, BE_information (Before Execution data) and AE_information (After Execution data). Job_Desc stores the detailed description of a job, such as job identifier, its type and its data size. The Job_Desc base class contains BE_Information class and AE_Information class. BE_Information stores the information about a job that can be collected by static analysis of the job. This information includes lines of code, number of parallel regions, and number of parallel loops. In certain cases, other information like nesting level of the parallel loops, upperbound and lowerbound of the index variables in a loop and shapes of the iteration spaces of the loops can be obtained before executing the job. In other cases, these data can only be collected *after execution*. In many cases though, these data vary from one execution to another. For such cases it is undesirable to use such data for program analysis.

AE_Information comprises data like total execution time, total instruction count, total number of floating point operations in the job, maximum number of threads used to execute the job etc. AE_Information contains information has been executed. Each resourceprovider_information class is described by resource provider type (workstation, cluster, SMP etc.), total number of processors, clock per instruction etc. Each execution environment is also associated with the analysis details for the fraction of the job that has been executed in the environment. These details are gathered through runtime performance analysis of the job. The class

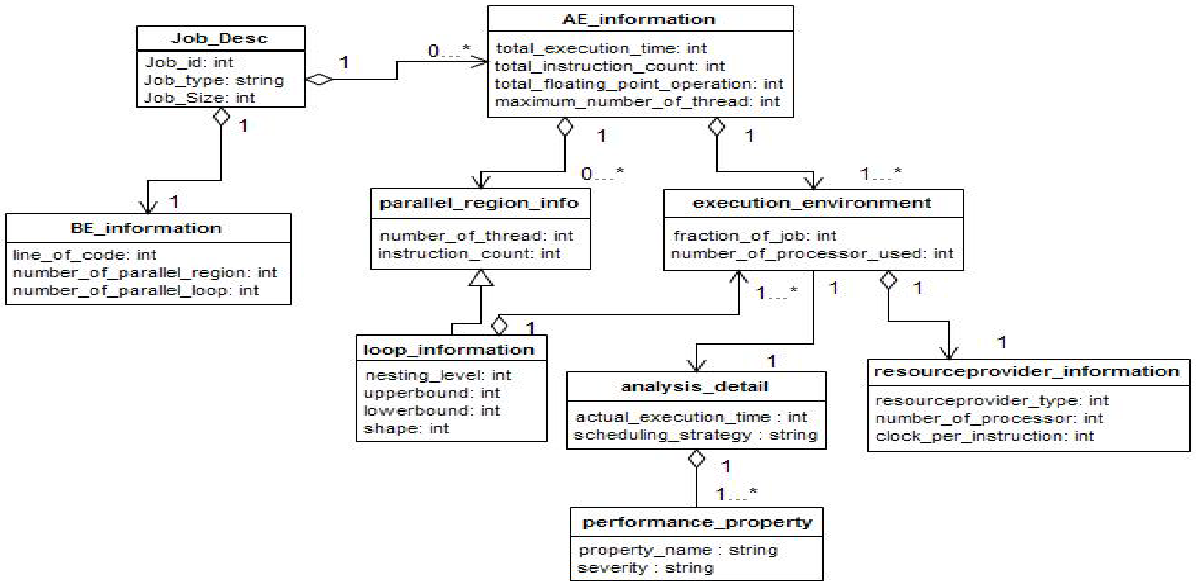


Figure 2: UML representation of the base classes of execution history of a job.

related to the parallel regions. We store information for a parallel region only if it is a significant region (i.e. it executes for a significant amount of time with respect to the total execution time of the job). However, information related to every parallel region can also be stored using the data model. The class parallel_region_info includes information like number of threads used to execute the region, instruction count for the region etc. A parallel loop must be a parallel region; therefore, loop_information (for parallel loops only) is a sub-class of parallel_region_info. It includes information on nesting level of the loop, upper bound and lower bound of the loop and the shape of its iteration space (triangular, square etc.). As mentioned before, some of this information can be collected through static analysis of the code. In many cases, though, these data vary from execution to execution and therefore such cases must be handled separately and are not within the scope of the current paper.

PRAGMA facilitates adaptive execution of jobs. Thus, execution environment of a job may change during its execution. Furthermore, in case of certain performance problems, a job may migrate from one resource provider to another resource provider. Therefore, fractions of a job may execute in different execution environments. Hence, a job may be related to different execution environments as depicted in Figure 2 (class execution_environment). Each execution environment is described by the fraction of a job executed in that environment and required number of processors to execute that fraction. Execution environment is associated with resource provider information (class resourceprovider_information) on which the fraction of the job

analysis_detail stores actual execution time for the loop fractions (if the significant parallel region is a loop), its scheduling strategy (static, dynamic), etc. For each fraction of the job executed in a specific execution environment, performance analysis is carried out on the basis of certain performance properties [2]. Thus performance_property is a derived class of analysis_detail that stores the severity values for every performance property during execution [2].

4.3 Proposed methodology

In our proposed framework for feedback guided job modeling, when a job runs in Grid environment with different QoS expectations, its execution information along with performance analysis details [18] are gathered as e*xecution history* of the job. The e*xecution history* is used as feedback information for future execution of the same job or similar kinds of jobs. Our endeavor is to build up a job model as precisely as possible without executing it provided that the *execution history* of the same job or similar type of job has already been stored in the database.

When a job arrives for execution, it is first checked whether a job of same type and of exactly same size has been executed earlier. In such case, the new job and the job stored before are considered to be the same jobs. On the other hand, two jobs of same type but with different data sizes are considered as similar jobs. In order to determine the type, merely the Job_type information stored in Job_Desc class is not sufficient. Job type must be related with the structure and other characteristics of the job. We are currently working with the techniques for identifying same type and similar type of jobs.

However, these techniques are not within the scope of this paper. Now let us explain our methodology for job modeling with an example. Consider two jobs, A and B. Let A has been executed earlier and as a result we have the *execution history* for job A. We have *BE* type of data for job B. Thus, there are three sets of data: *BE* type data for A, *AE* type data for A and *BE* type data for B. From the three sets of data we make a prediction of the *AE* data of job B based on whether A is same as B or is similar to B. This prediction of *AE* data can be used to estimate the resource requirement for a job. In addition, the analysis details of job A can be used to improve the performance of job B. Consider A and B are two similar types of jobs. Thus, from the *execution history* of job A we can predict a suitable execution environment for B. When a job executes for the first time, there cannot be any after execution information. A job may have more than one instances of AE_Information as it might have been executed on different machines.

5. Examples

In this section we demonstrate our proposed techniques taking example codes in parallel Java applications. Minor modifications have been made to the original source codes in order to run them in PRAGMA environment. Parallel jobs are implemented using Java OpenMP, i.e. JOMP. Three different programs are considered in this paper:

Matrix Multiplication: The conventional matrix multiplication code consists of one nested loop of depth three. The outer most loop is a significant loop and the shape of the loop is square. Parallelization of the significant loop is done using Java OpenMP directive. Data sizes in the experiments have been varied from 1000x1000 to 5000x5000.

Gaussian Elimination (Linpack benchmark): This program provides solution for a system of linear equations, Ax=B, where *A* is a known matrix of size *N*X*N*, *x* is the required solution vector, and *B* is a known vector of size *N*. The major significant loop is triangular in shape and parallelization of this loop is done using JOMP. This benchmark is experimented with varying data size from 1000 to 6000.

LU Matrix Factorization (SciMark 2.0 benchmark): This program computes LU factorization followed by a triangular solve of a dense NxN matrix using partial pivoting and dense matrix operations. This major significant loop in this test code is triangular in shape. Parallelization of this significant loop is done using JOMP directive. For the experimental purpose, data sizes have been varied from 1000 to 8000.

Table1: Before Execution analysis details.

Job	Input Data size	LoC	Number of Parallel Loop	Parallel Loop Details	
				Nesting Level	Shape
Matrix Multiplication	2000X2000	109	1	3	Square
	3000X3000	109	1	3	Square
	4000X4000	109	1	3	Square
	5000X5000	109	1	3	Square
Gaussian Elimination	2000	150	1	2	Triangular
	4000	150	1	2	Triangular
	5000	150	1	2	Triangular
	6000	150	1	2	Triangular
LU matrix factorization	2000X2000	108	1	3	Triangular
	4000X4000	108	1	3	Triangular
	6000X6000	108	1	3	Triangular
	8000X8000	108	1	3	Triangular

Table 1 shows *Before Execution (BE)* data for the three above mentioned programs. The columns in the table hold identification of each program, input data size, Line of Code (LoC), number of parallel loops, nesting level of the loops and shape of the loops.

Initially, the above mentioned three jobs with different data sizes are executed within the PRAGMA environment [2] and their execution details are stored in the *execution history*. We have implemented only three tuning strategies - changing the number of threads, changing the local scheduling strategies and migrating the job if necessary. In the current example, only by changing the number of threads while executing the job, desired performance could be achieved by the client. The analysis details [18] contain actual execution time of the job fraction, number of threads (processors) that have been used to execute the job fraction, scheduling strategy that has been used for local scheduling and total execution time of the job. From the *execution history*, it is now easy to determine the best execution environment for each job. Table 2 provides the best execution environments that are perceived from *execution history* of the three jobs with known data sizes. It also provides total execution times for all above mentioned jobs when they are executed within best execution environment. For example, we perceived the best execution environment for Matrix Multiplication of data size 4000X4000, when it has been executed with 5 processors using static scheduling strategy on HP server or the best execution environment for LU matrix factorization of data size 6000X6000, when it has been executed on HP server with 6 processors using dynamic scheduling strategy. Table 2 also depicts resource provider information on which the jobs were executed and achieved the warranted performance.

Table 2: Best execution environment of jobs perceived from execution history.

Job	Job_Size	Execution		Total Execution Time (sec)	Resource Provider Details
		Number of Threads	Scheduling Strategy		
Matrix Multiplication	2000X2000	4	Static	62	HP ProLiant ML570 G4 with SMP 4 Intel Core2 Duo Processors of 3.2 GHz with 8 MB cache and 8 GB memory (referred as HP Server) Operating System: Suse Linux(Kernel release 2.6.16.21-0.8-bigsmp
	3000X3000	4	Static	229	
	4000X4000	5	Static	558	
	5000X5000	6	Static	1044	
Gaussian Elimination	2000	2	Dynamic	13	
	4000	3	Dynamic	82	
	6000	4	Dynamic	85	
LU matrix factorization	2000X2000	3	Dynamic	10	
	4000X4000	4	Dynamic	84	
	6000X6000	6	Dynamic	188	
	8000X8000	8	Dynamic	360	

Table 3: Sample execution history.

Job	Job_Size	Total Execution Time(ms)	Execution Environment		Resource Provider Details
			Number of Thread	Scheduling Strategy	
Matrix Multiplication	1000X1000	5827	4	Static	HP ProLiant ML570 G4 with SMP 4 Intel Core2 Duo Processors of 3.2 GHz with 8 MB cache and 8 GB memory (referred as HP Server) Operating System: Suse Linux(Kernel release 2.6.16.21-0.8-bigsmp
	1500X1500	19500	4	Static	
	2500X2500	119640	4	Static	
	3500X3500	399933	4	Static	
Gaussian Elimination	1000	2204	2	Dynamic	
	1500	6057	2	Dynamic	
	2000	28838	2	Dynamic	
LU matrix factorization	1000X1000	1332	3	Dynamic	
	1500X1500	4490	3	Dynamic	
	2000X2000	10763	3	Dynamic	

Next, two experiments have been carried out as described below.

Experiment 1: Running same job in PRAGMA environment

When a parallel job is submitted to PRAGMA Environment, PRAGMA applies its feedback guided job modeling strategy and identifies that the newly submitted job is same as one of the previously executed jobs. Next, it retrieves the *execution history* of the previously executed jobs. On the basis of the history, the best execution environment is selected for the new job. Thus when the same jobs are submitted again, they are executed in the best execution environment retrieved from the *execution history*. For example, If Matrix Multiplication of size 4000X4000 is submitted again in PRAGMA, feedback guided job modeling recognizes the job as same job in comparison with the jobs already executed earlier. The best execution environment for Matrix Multiplication of size 4000X4000 is retrieved and the job is executed in the best execution environment. Similarly, if a LU matrix factorization of 6000X6000 is submitted again, it will be executed in the best execution environment with 6 processors, dynamic scheduling strategy. Figure 3, Figure 4 and Figure 5 compare the execution time of same jobs in the existing PRAGMA environment and in PRAGMA environment with feedback guided job modeling technique for Matrix Multiplication, Gaussian Elimination and LU matrix factorization respectively.

Experiment 2: Running similar jobs in PRAGMA environment

In case of similar jobs, consider initially we have the *execution history* of the three jobs (or job fractions) executed in a single execution environment. Table 3 shows a sample *execution history* that contains execution times, execution environments, resource provider details for each of the three above mentioned jobs with specific job sizes. The execution time of a similar job (in this case, same jobs with different input data sizes) is predicted from the *execution history*. Prediction is done here on the basis of mean and linear regression of the previous execution times. Thus, when a new job is submitted in PRAGMA environment, if it is found that the same job was never executed, but a similar job[1] was executed previously, PRAGMA consults its *execution history.* Then using linear regression, PRAGMA predicts the execution time of the new job and considers the predicted value as its expected completion time (ect) [2]. Thus, expected completion times (ect) of Matrix Multiplication of sizes 2000X2000, 3000X3000, 4000X4000, 5000X5000 are predicted using mean and linear regression in a constant execution environment (HP server, 4 processors, static scheduling strategy). The Matrix Multiplication jobs of sizes 2000X2000, 3000X3000, 4000X4000, 5000X5000 are submitted with the predicted execution time as expected completion time (ect) to the same execution environment in PRAGMA. Similarly, expected completion time of Gaussian Elimination of sizes 2500, 3000, 5000 are predicted in a constant execution environment (HP server, 2 processors, dynamic scheduling strategy) and the predicted execution times are used as expected completion times at the time of initial job submission. In case of LU matrix factorization of sizes 2500X2500, 3000X3000 and 5000X5000, same techniques are repeated and the predicted execution times are used as the expected completion times during their initial submission. In the PRAGMA environment, each job achieves warranted performance and completes its execution within the expected completion time that has been predicted using *execution history* data. Figure 6, Figure 7 and Figure 8 compare the execution times of a job in PRAGMA (when job is initially submitted with only one processor (default) with a randomly selected expected completion time) [[1],[18]] and the execution times of a job when the job is submitted in PRAGMA environment with feedback guided job modeling technique where the predicted execution time is taken as the expected completion time for Matrix Multiplication, Gaussian Elimination and LU matrix factorization.

1 The technique for deciding whether a job is *similar* to a previously executed job is not within the scope of this paper.

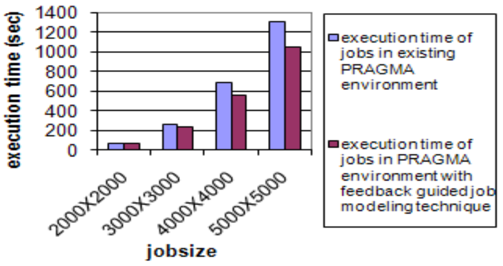


Figure 3: Comparison between the execution times of Matrix Multiplication in existing PRAGMA environment and in PRAGMA environment with feedback guided job modeling technique.

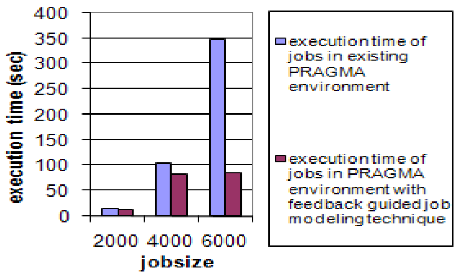


Figure 4: Comparison between the execution times of Gaussian Elimination in existing PRAGMA environment and in PRAGMA environment with feedback guided job modeling technique.

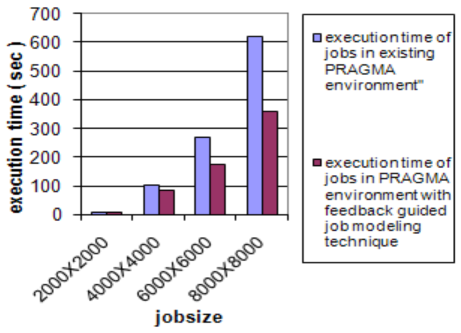


Figure 5: Comparison between the execution times of LU matrix factorization in existing PRAGMA environment and in PRAGMA environment with feedback guided job modeling technique.

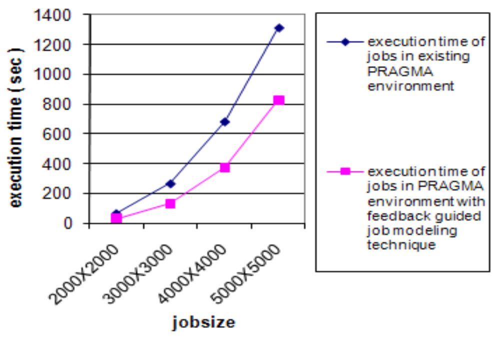


Figure 6: Comparison between the execution times of Matrix Multiplication in existing PRAGMA environment and in PRAGMA environment with feedback guided job modeling technique.

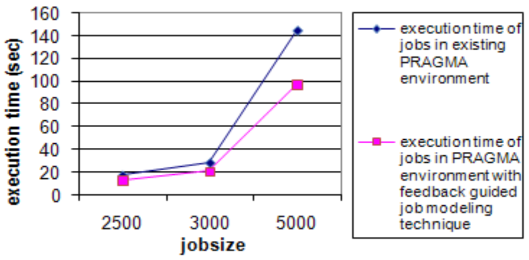


Figure 7: Comparison between the execution times of Gaussian Elimination in existing PRAGMA environment and in PRAGMA environment with feedback guided job modeling technique.

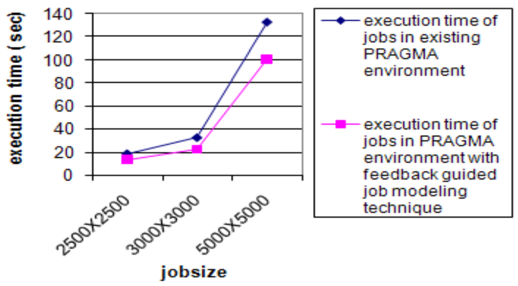


Figure 8: Comparison between the execution times of LU matrix factorization in existing PRAGMA environment and in PRAGMA environment with feedback guided job modeling technique.

6. Conclusion and Future Work

In this paper we have proposed a framework for managing execution information of jobs that are used as the feedback data and this feedback data are stored as *execution history* of jobs. The *execution history* is used later to predict expected completion time and the resource requirements of a newly submitted job in PRAGMA environment. Sometimes the *execution history* is used to improve the execution performance of the newly submitted job. The feedback guided job modeling provides a basis of finding job requirements' before its execution without common users' intervention. Besides discussing the technique for managing job execution information using feedback guided job modeling in PRAGMA environment, the automatic job modeling module within MAS [1] has also been discussed here. We are improving our existing feedback guided job modeling module to make it a fully automatic job modeling module in near future.

Acknowledgement

This research work has received support from the project entitled "Developing Multi-Agent System for Performance Based Resource Brokering and Management in Computational Grid Environment" funded by Department of Science and Technology, Government of India under the SERC scheme.

References

[1] Roy S., N. Mukherjee: "Adaptive Execution of Jobs in Computational Grid Environment", Published in Journal of Computer Science and Technology, Springer, vol.24, No.5, September 2009.
[2] Roy S., M. Sarkar, N. Mukherjee: "Optimizing resource allocation for multiple concurrent jobs in grid environment", Published in ICPADS 2007: pp. 1-8.
[3] Christodoulopoulos K., V. Gkamas, E. A. Varvarigos: "Statistical Analysis and Modeling of Jobs in a Grid Environment", Published in the Journal of Grid Computing (2008) 6:77–101.
[4] Li H., R. Heusdens, M. Muskulus, L. Wolters: "Analysis and Synthesis of Pseudo-Periodic Job arrivals in Grids: A matching Pursuit Approach" in Proceedings of CCGrid'07.
[5] Li H., M. Muskulus, L. Wolters: "Modeling Job Arrivals in a Data-Intensive Grid" in Proceedings of the 12th JSSPP'2006 (2006).
[6] Real Time Monitor: http://gridportal.hep.ph.ic.ac.uk/rtm/
[7] http://www2.epcc.ed.ac.uk/computing/research_activities/jomp/grande.html
[8] Kuntraruk J., W. M. Pottenger, A. M. Ross: "Application Resource Requirement Estimation in a Parallel-Pipeline Model of Execution", in IEEE Trans. Parallel Distributed System 16(12): 1154-1165 (2005).
[9] Adve V. S.: "Analyzing the Behavior and Performance of Parallel Programs", in Computer Sciences Technical Report #1201 for the degree of Doctor of Philosophy of University of Wisconsin-Madison, December 1993.
[10] The EGEE project homepage: http://public.eu-egee.org/
[11] Kuntraruk J., W.M. Pottenger: "Massively Parallel Distributed Feature Extraction in Textual Data Mining Using HDDITM", in Proceedings of the Tenth IEEE International Symposium on High Performance Distributed Computing, August 2001.
[12] Ali A., A. Anjum, J. Bunn, R. Cavanaugh, F. van Lingen, R. McClatchey, M. A. Mehmood, H. Newman, C. Steenberg, M. Thomas, I. Willers: "Predicting the Resource Requirements of a Job Submission. In: Computing in High Energy Physics", 2004, Interlaken, Switzerland.
[13] Bohlouli M., M. Analoui: "Grid-HPA: Predicting Resource Requirements of a Job in the Grid Computing Environment": in Proceedings of World Academy of Science, Engineering and Technology, August, 2009.
[14] Nemeth Z., V. Sunderam: "A Comparison of Conventional Distributed Computing Environments and Computational Grids", in Proceedings of the International Conference on Computational Science-Part II (ICCS '02), LNCC, vol. 2330, pp. 729-738, April 2002.
[15] http://www.csse.monash.edu.au/~davida/nimrod/nimrodg.htm
[16] Globus, web site: http://www.globus.org
[17] Roy S.: "Performance-based Resource Management in Computational Grid Environment". Thesis for the degree of Doctor of Philosophy of Jadavpur University, Kolkata, October 2007.
[18] De Sarkar A., S. Roy, D. Ghosh, R. Mukhopadhyay, N. Mukherjee: "An Adaptive Execution Scheme for Achieving Guaranteed Performance in Computational Grids", Published in the Journal of Grid Computing, ISSN: 1570-7873 (Print) 1572-9814 (Online), vol. 8, pp. 109-131, 2010.

Skeleton/Pattern Programming with an Adder Operator for Grid and Cloud Platforms

J. Villalobos, B. Wilkinson
Department of Computer Science, University of North Carolina, Charlotte, NC, USA

Abstract— *Pattern operators are extensions to the Pattern/Skeleton parallel programming approach used to apply two types of communication patterns to the same data. The operators are intended to simplify the wide range of possible patterns and skeletons. The abstraction helps manage non-functional concerns on the Grid/Cloud environments. This paper explains how the pattern operators work on synchronous cyclic undirected graph patterns, and it shows examples on how they are used. A prototype was created to test the feasibility of the idea. The example used to show the operator approach is the addition of termination detection to a discrete solution to a PDE. The example can be coded with 27.31% less non-functional code than a similar implementation in MPJ, and its programmability index is 13.5% compared to MPJ's 9.85%. The overhead for an empty pattern with low communication was 15%. The use of pattern operators can reduce the number of skeletons/patterns developed.*

Keywords: Skeletons, patterns, grid, cloud, operators

1. Introduction

The advent of Grid computing during the past decade has created a heterogeneous computing environment where the users of computing resources have been exposed to multiple types of architectures, network topologies, security issues and other non-functional concerns. These challenges need to be addressed. We are using the term heterogeneous environment to refer to the Grid and to cloud computing. Some aspects of cloud computing require the use of abstractions such as skeletons/patterns (SPs), particularly the need to separate the resources of the cloud from the users and developers. The main problem at hand is how to program for this heterogeneous environment given all the possible variations that may need to be accounted for in order to run a single program in different configurations efficiently. The approach we favor is the use of SPs.

Skeletons are directed, acyclic graphs, and their implementations are data parallel. The algorithms create a flow of data that streams from one stage to the other until the necessary algorithms have been performed to it. Patterns are cyclic, undirected graphs. They are data-parallel and they require synchronization during the execution. The start and end of the patterns are the same as the skeleton, that is, they start with some mapping of the initial data to the nodes, and then end by converging the processed data into a sink node. Figure 1 shows the common life cycle for both skeletons and patterns. The most recurring skeletons are map, reduce, workpool/farm, and pipeline. The most recurring patterns are stencil and all-to-all. More complex skeletons like divide-and-conquer can be constructed from simpler skeletons through nesting [1] [2]. The patterns have a wider set of non-standard pattern types that cannot be made from a basic set of patterns as is the case with skeletons.

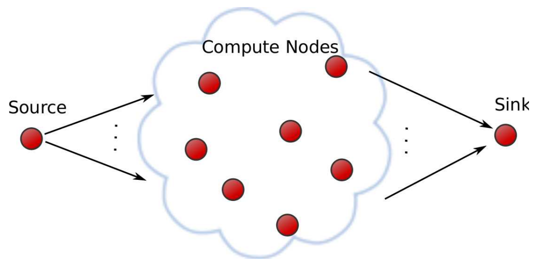


Fig. 1: Basic skeleton pattern organization.

There are three advantages to using SPs over the industry standards today (MPI [3], OpenMP [4], and explicit thread libraries). The first is that SPs hide deadlock and race conditions from the user. They provide implicit parallelization to the user programmer. This is done by giving the user programmer an interface. When the user programmer gives the implementation to a framework, the framework will run the interface while taking care of the race conditions and deadlock. The user is never aware of the problem. The second advantage is a reduction in code and development time. Macdonald et al. showed that coding with SPs requires less coding, and is simpler in comparison to MPI [5]. Aldinucci et al. also have shown frameworks such as Lithium and Muskel that use object-oriented Java to provide the benefits of skeletons to the user programmer [1]. Object-oriented languages have created the abstraction that is necessary for the concepts behind SPs to be provided in a form that is simpler to understand. Previously, projects such as eSkel [6], Sketo [7], and DPnDP [8] provided skeletons using procedural C/C++ (which is not typed) but they create an environment where mistakes are hard to find [8]. The third advantage for SPs came in with the increased need to abstract the parallelization away from the computational resources as is needed in cloud and Grid computing. The abstraction is needed mainly in Grid computing because the environment is made up of multiple architectures and network topologies. Ideally, service providers want an application to run on this environment while minimizing the amount of knowledge the user programmer needs to code it. In the case of cloud computing, the environment tends to be more homogeneous and controlled by a single entity, but the service provider also wants to provide the computation resources in a way that they can optimize the use of the hardware. The hardware optimization leads to servicing more customers. SPs are a pertinent option to abstract the use of the resources because they allow the user programmer to code the problem using the provided API, and they give the Grid/cloud maintainers space to manage the non-functional requirements. The API and interfaces, in effect, create an extra layer between the hardware and the user programmer. One could argue that skeletons have started to be introduced into the cloud; MapReduce can be cited as a successful example of skeleton use in the cloud environment [9]. More skeletons are needed in the future since MapReduce is not optimal for all types of parallel programming algorithms.

Despite the benefits, SPs have some persistent drawbacks. In his manifesto, Cole explains that SPs must "show the pay-back", and he stresses simplicity [6]. However, many still believe that the number of SPs needed to address parallel computing is infinite [10]. This has some implications; the user programmer may be faced with a library of SPs so large that he is discouraged from using the approach. On the other extreme, the user programmer, despite having multiple SPs to choose from, does not find the pattern that he needs for the problem, and therefore is tempted to just use lower level tools. One can intuitively surmise that there can be a basic set of SPs from which all the other SPs can be created. This could be possible by features such as nesting. In nesting, the user programmer is able deploy new SPs from inside the SPs, which allows for an exponential increase of SPs without increasing the size of the basic set. Nesting also allows for the use of libraries that contain SPs themselves. With some operators and a basic set, one could see a Turing-complete (so to speak) set of patterns from which all possible parallel programs can be created.

Section 2 presents a high level explanation about pattern operators. Subsection 2.1 presents an example using Java. It explains the use of interfaces to create computation modules and the interfaces used to create data containers. Subsection 2.2 presents some important details on implementing SPs into a framework we call the Seeds framework. Subsection 2.3 presents the implementation of the adder operator in Seeds using the tools explained in Section 2.2. Section 3 presents the results from measuring the adder operator on the dimensions of performance and programmability. Finally, Section 4 mentions the related work.

2. Pattern Operators

Pattern operators are elements that work on the same piece of data. We present here the addition operator for synchronous patterns. The creation of this operator comes about to address stateful algorithms such as discrete solutions to PDE's and practical solutions to particle dynamics algorithms. In these types of algorithms, there are different communication patterns at different stages of programming. In the example of a discrete simulation of heat distribution, multiple cells on a stencil pattern work in a loop parallel fashion, computing and synchronizing on each iteration. However, every x iterations, they must implement an all-to-all communication pattern to run an algorithm to detect termination. That is used to check if all cells have converged on a value and all the cells should at that point stop computing. Figure 2 shows the example of this approach.

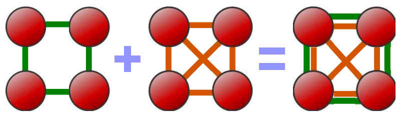


Fig. 2: Adding a Stencil plus an All-to-All synchronous pattern.

Similarly, the practical approach to solving particle dynamics combines multiple communication algorithms. Let us first review a simple version of particle dynamics. In this version, an all-to-all communication pattern is run on every iteration and the information for every particle is used to calculate the future momentum of each particle. This has O(N^2) complexity. Another algorithm has a lower complexity, but its implementation is to use a stencil where the particles will calculate its momentum based on n of its closes neighbors. This stencil pattern is performed for x iterations. Every x iterations, the algorithm switches into an all-to-all pattern of communication to update all particles and reduce the error that the algorithm inevitably accumulates. With just these two examples, it is easy to see that many more algorithms fall into this category where one has multiple layers of communication patterns that work on the same data. In the example of heat distribution, the data is a set of pixels that represent the heat energy present at that point. In the case of particle dynamics, the data represents momentum for each particle at that instant in time.

2.1 Example

Figures 3 and 4 show an example where an all-to-all termination detection algorithm is used to determine if there is convergence after performing a stencil algorithm for some number of iterations. A discrete approach to the problem of heat distribution was used to test the code shown in the figures. Figure 3 shows the code used to create the algorithm for heat distribution. Some of the problem-specific code was omitted in the interest of brevity. The class HeatDistribution extends a Stencil abstract class. This requires the user programmer to implement some signature methods. The Javadoc for each signature method is used to instruct the user programmer on the purpose of each method and their interaction within the framework. The DiffuseData() method is used to get the segments of data from the user programmer. GatherData() is used to get the processed segments of data back from the user. OneIterationCompute() is used as the main computation method. Because the algorithms are loop-parallel and the framework needs to gain back control in order to organize multiple patterns, the user is instructed the method should only run one iteration of the main loop in the application. initializeModule() is used to allow the user programmer to pass string arguments to the remote instantiation just after the modules get initialized.

```
public class HeatDistribution extends Stencil {
 private static final long serialVersionUID = 1L;
 int LoopCount ;
 public HeatDistribution(){
  LoopCount = 0;
 }
 @Override
 public StencilData DiffuseData(int segment) {
  int w = 10, h = 10;
  double[][] m = new double[10][10];
  /** init matrix m with file or user input*/
  HeatDistributionData heat = new
             HeatDistributionData(m, w, h);
  return heat;
 }
 @Override
 public void GatherData(int segment, StencilData dat) {
  HeatDistributionData heat = (HeatDistributionData) dat;
    /** print or store results*/
 }
 @Override
 public boolean OneIterationCompute(StencilData data) {
  HeatDistributionData heat = (HeatDistributionData) data;
  double[][] m = new double[heat.Width][heat.Height];
  /** compute core matrix */
  /** compute sides (borders)*/
  /** compute corners */
  /** set if this node is done*/
  heat.matrix = m;
  return false;
 }
 @Override
 public int getCellCount() {
  return 4; //four nodes for this example
 }
 @Override
 public void initializeModule(String[] args) {
  /*not used*/
 }
}
```

Fig. 3: HeatDistribution class extends Stencil and fills in the required interfaces.

Figure 4 shows the TerminationDetection class, which extends CompleteSyncGraph. Similar to the stencil pattern, CompleteSyncGraph also requires some signature methods. The pattern has a DiffuseData() and GatherData() method but they are not used for this example since the second pattern in the operator is used for its computation function only. getCellCount() is the number of processes needed for

```
public class TerminationDetection extends CompleteSyncGraph{
 @Override
 public AllToAllData DiffuseData(int segment) { // not used
  return null;
 }
 @Override
 public void GatherData(int segment, AllToAllData data) {
  // not used
 }
 @Override
 public boolean OneIterationCompute(AllToAllData data) {
  HeatDistributionData d = (HeatDistributionData) data;
  return d.Terminated;
 }
 @Override
 public int getCellCount() { // not used really
  return 4;
 }
 @Override
 public void initializeModule(String[] args) {/*not used*/}
}
```

Fig. 4: Termination detection using all-to-tall pattern.

the computation and must return the same number on both patterns so that communication patterns fit together.

Figure 5 shows the main data object used for both the patterns. The main advantages sought in using the pattern adder is to provide the user programmer with the ability to have two communication patterns work on the same data. Our approach to patterns has the requirement of having all information used for communication travel in the form of serializable objects. Additionally, the stencil pattern adds other signature methods that are needed in order to control the communication on behalf of the user programmer. CompleSyncGraph also adds signature methods. HeatDistributionData implements both StencilData and AllToAllData so that it can be handled by both patterns.

Both of these modules are inserted into the framework using a bootstrapping executable class. Figure 6 shows the executable the user programmer implements in order to add the stencil pattern plus the CompleteSyncGraph pattern. The two are added using an Operand class, which is used to hold together three characteristics each pattern needs, which are: the initialization arguments if any, the host anchors if any, and an instance of the pattern's module. Anchors are used to tie a special node the host that has to run it. The main use for the anchor is to specify where the source and sink nodes are to be run, since they usually have to be where the data is. The executable class also has some code to start the framework and shutdown the framework, which will self-deploy. After creating the operands, the pattern-adder operator is deployed by starting a new pattern called an AdderOperator. The framework, by default, will spawn and monitor the new pattern on a separate thread. The user programmer can just wait for the pattern to complete using waitOnPattern() method.

```
ublic class HeatDistributionData
        implements StencilData , AllToAllData {
 boolean Terminated;
 public double [][] matrix;
 public int Width,Height;
 SyncData[] Sides;
 public HeatDistributionData(double [][]m
                        , int width , int height){
 }
 /** Stencil data signature methods */
 @Override
 public Data getBottom() {}
 @Override
 public Data getLeft() {}
 @Override
 public Data getRight() {}
 @Override
 public Data getTop() {}
 @Override
 public void setBottom(Data data) {}
 @Override
 public void setLeft(Data data) {}
 @Override
 public void setRight(Data data) {}
 @Override
 public void setTop(Data data) {     }
 /** The All-to-All Data signature methods */
 @Override
 public Data getSyncData() {/**return data for all*/}
 @Override
 public void setSyncDataList(List<Data> dat) {
    /** get data from all */
 }
}
```

Fig. 5: The main data extends both the StencilData and AllToAllData. The object is used to hold the state-full data for the main processing loops.

```
public class RunHeatDistribution {
 public static void main(String[] args) {
  Deployer deploy;
  try {
   Seeds.start("/path/of/shuttle/folder/pgaf", false);
    /**first pattern */
    Operand f = new Operand(
        (String) null
        , new Anchor("Kronos", DataFlowRoll.SINK_SOURCE)
        , new HeatDistribution() );
    /**second pattern*/
    Operand s = new Operand(
        (String) null
        , new Anchor("Kronos", DataFlowRoll.SINK_SOURCE)
        , new TerminationDetection() );
    /**create the operator*/
    AdderOperator add = new AdderOperator(
        new ModuleAdder( 100, f, 1, s )   );
    /**start pattern and get tracking id*/
    PipeID p_id = Seeds.startPattern( add );
    /**wait for pattern to finish*/
    Seeds.waitOnPattern(p_id);
   Seeds.stop();
  } catch (Exception e) {
    /**catch exceptions*/
  }
 }
}
```

Fig. 6: RunHeatDistribution is used to create the operator and start the pattern.

2.2 Implementation Details

The SPs are divided into unstructured and structured. The unstructured patterns are the templates such as the map, reduce, and workpool/farm patterns. They are referred to as unstructured because the number of processes can change dynamically. A workpool/farm can work with 10 nodes the same as it works with 1000 nodes without the need to modify its implementation or having to add special code to the framework to handle the change. These are features that are inherent in the definition of these three skeletons. The other algorithms that require a specific number of processes are the pipeline skeleton, the synchronous loop-parallel patterns such as the stencil and its modifications, and the complete graph or all-to-all pattern.

Our Seeds framework implements the differences by using UnorderedTemplate for the unstructured skeletons, and OrderedTemplate for the structured SP's. Furthermore, each template is inherited to implement the specific pattern. PipeLineTemplate inherits OrderedTemplate to implement the skeletons. Figures 7 and 8 show the UML for the different classes that inherit OrderedTemplate and UnorderedTemplate respectively.

Map and reduce skeletons are not implemented. There is a template created for convenience called LoaderTemplate. LoaderTemplate inherits UnorderedTemplate and its goal is to load some initial data into the computation units for OrderedTemplate algorithms. Once the computation units are loaded, they can start working on one of the OrderedTemplate implementations. At the end of the OrderedTemplate SP, control is returned to LoaderTemplate, which sends back the data to the sink node. The data may have been modified during the computation as it would happen when implementing a stencil, or the data may not have any significance once the job is complete.

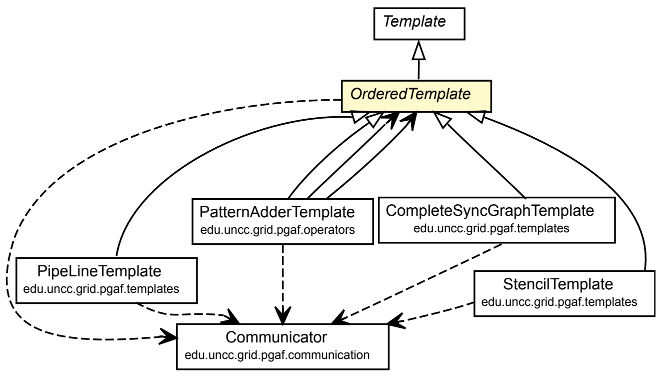

Template
OrderedTemplate
PatternAdderTemplate
edu.uncc.grid.pgaf.operators
CompleteSyncGraphTemplate
edu.uncc.grid.pgaf.templates
PipeLineTemplate
edu.uncc.grid.pgaf.templates
StencilTemplate
edu.uncc.grid.pgaf.templates
Communicator
edu.uncc.grid.pgaf.communication

Fig. 7: The OrderedTemplate needs a specific number of processes. The Stencil and CompleteSyncGraph inherit this class.

In order to provide structure to the user programmer, an interface is created using abstract classes that inherit the BasicLayerInterface abstract class. This is used to provide the user programmer with the signature functions that must be implemented in order to successfully interact with the framework. The interface is like a form presented to the user with instructions on how it should be filled in. Figure 9 shows the UML diagram for the BasicLayerInterface class and some interfaces that inherit it.

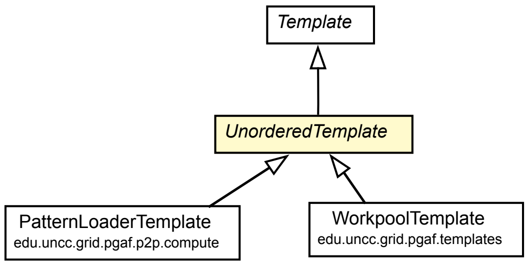


Fig. 8: Template class. OrderedTemplate and UnorderedTemplate extend it.

The basic layer: If the user wants to implement some skeleton or pattern, he needs to create at least three classes: the module that implements an SP interface, at least one class that implements a Data interface, and an executable class that connects the module to the framework, and deploys the framework if it is not running already. An example of this procedure can be seen in Section 2.1.

The advanced layer: This advanced layer user is interested in implementing a pattern that cannot be implemented with the existing skeletons/patterns, or he wants to test a new optimization method, or he wants to provide a pattern that is more convenient to use than the options already offered by the framework. In this case, the user needs to implement at least two classes, and one interface. The advanced user must decide if the SP needs a structured or unstructured implementation, a class that inherits one or the other must be created. If an unordered template is created, only one class is needed. If the SP is structured, then the advanced user must also implement a LoaderTemplate. The advance user may see the need to create Data interfaces to be able to steer the user programmer into the communication patterns that are required by the advanced user's SP; however, this is optional.

The expert layer: The expert layer is the machine room - it has many primitive data structures and behavioral patterns that look like an MPI implementation. The objects that the expert layer provide to the layer above tend to be complicated because the heterogeneity of the environment. It has long if statements that account for each type of network the nodes might be in, and each type of memory management system potentially available. The expert layer is only for Grid computing experts and parallel programming researchers.

2.3 The Operators

The operators at present are only implemented for the OrderedTemplate SPs because only the addition operator seems to be beneficial in reducing SPs complexity for the user programmer. Future endeavors may include adding operators to unstructured skeletons, to get similar benefits as we show can be had from the addition operator.

The operators are implemented by inheriting OrderedTemplate. Once this OrderedTemplate is loaded, OperatorTemplate runs the first SP to load the initial computational units. Then it enters into the main loop-parallel cycle. It run the first operand for n iterations, and then it runs the next SP for x iterations until either one of the SPs return true. The computation for these SPs return false if the program is not done computing. When the main loop-parallel cycle is done, the operator pattern returns the processed data units to the first operand's GatherData() method. The operator implements LoaderTemplate to load and unload the initial data from the first SP. The second SP only contributes the computation operation, and the Diffuse/Gather operations are ignored for this SP. Figure 9 shows a diagram that describes most of the interaction among the classes that happens when running the operator template. The small tabs inside the square are used to mention the BasicLayerInterface class that is used by the Template class. For example: Stencil class inherits BasiclayerInterface, and it is used by StecilTemplate class. Together, both classes implement a stencil pattern. The two patterns are added into the PatternAdder interface, which is then executed by the AdderTemplate. Because the adder template is an OrderedTemplate, it must specify a PatternLoader interface, which is the PatternAdderLoader class. PatterAdderLoader inherits PatternLoader interface, and it is executed by the PatternLoaderTemplate. Finally, Seeds can execute the LoaderTemplate directly because it is an UnorderedTemplate. All templates implement a function for the client side node and one function for the server side node. The server side corresponds to the source and sink nodes, and the client side corresponds to the compute nodes.

3. Results

Tests were performed to validate the pattern adder operator. The two main concerns for extensions to the SP programming approach are the performance impact created by the extension, and programmability of the extension. In order to measure the performance overhead created by the pattern adder operator, we implemented a simple algorithm that uses both the stencil pattern and the complete pattern. The algorithm is trivial; it consists of sending a long integer type to the neighbor processes in the stencil pattern, and it repeats the process for the complete pattern. We consider this an empty grain size pattern. The time to run through one iteration is measured for the stencil pattern and for the complete pattern. The time taken to run the process is also measured for the pattern adder operator. The overhead is the difference between the pattern operator's time and the stencil plus the complete pattern's time. Figure 10 shows the result of this test. The test was performed on a 16-core

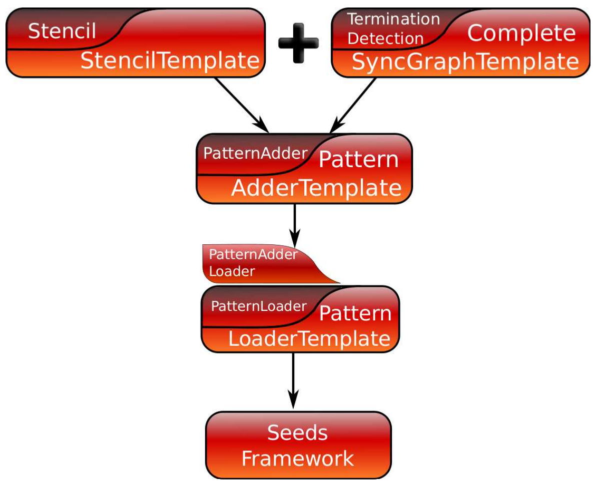


Fig. 9: Interface + Template pairs are drawn on the same square. The diagram shows the hierarchical interaction between the classes in order to execute a PatternAdder operator.

Xeon 2.93GHz server with 64GB memory. The number of processes is varied from 4 to 16. All the communications for this experiment were through shared memory. The results show that the overhead goes down as more processes are used for the computation. This is in part because the increasing communication overhead helps mask the overhead due to the operator. The overhead in comparison to an empty grain size is 15%, so grain size has to be adjusted to justify the use of the operator. The network speed also has an effect on the overhead. As Figure 10 shows, the increase in communication overhead reduces the overhead incurred due to the operator. The same test was done on a cluster of 3 dual-CPU Xeons (3.4GHz) with 8GB memory running four threads per server. The overhead for this test on an empty grain size pattern was 0.03% for nine processes. The network used was a Gigabit Ethernet.

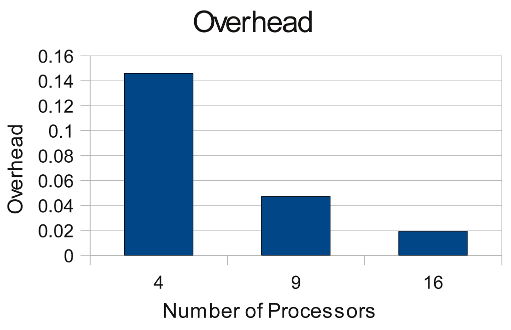


Fig. 10: Operator overhead measured on a shared-memory multi-core server.

Next, we measured the programmability. For this test we implemented the heat distribution algorithm using MPJ-Express. We also implemented the problem using the Seeds framework, and a serial version of the problem was used as control. The lines of code (LOC) were tagged on each implementation with the tags: functional, non-functional, automatic, comment, and log. The main tags are:

- Functional: The code that is dedicated to solving the problem. Most of the LOC in the serial implementation are considered functional.
- Non-functional: Is code primarily written to organize parallel processes and communications. MPJ-Express, and Seeds include code of this type to solve the problem in parallel.
- Automatic: This code is generated code by the IDE. Eclipse was used for the test. The generated code includes the class declaration, import lines, package declaration, and interface signature functions. Setters and getter can also be included as automatic code.

The programmability index is defined as:

$$\frac{\text{functional}}{\text{functional} + \text{non-functional}} \tag{1}$$

A higher ratio means a program with less non-functional code, and therefore we assume more readable and parallelized in less time. Figure 11 shows the result from this assessment. Seeds reduces the number of non-functional code by 27.31% over MPJ implementation. MPJ's programmability index for this implementation is 9.85%, and Seed's programmability index is 13.50%.

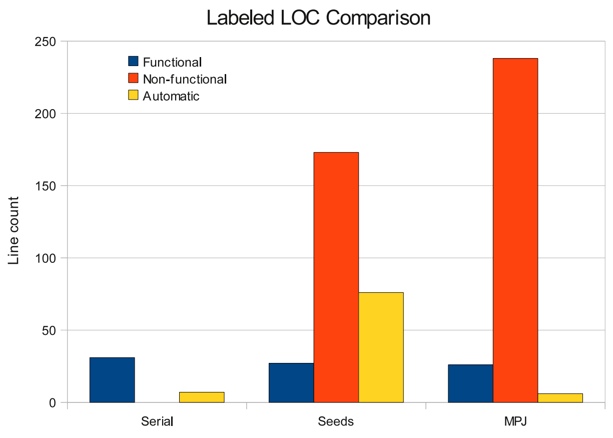


Fig. 11: the y axis shows the number of lines of code for each implementation. The LOC were counted for the serial implementation as well as for the Seeds and MPJ implementation.

4. Related Work

As referenced in Section 1. our work builds upon several frameworks designed to implement skeletons, patterns or both. Cole proposed a series of guidelines that must be met in order to create feasible skeleton interfaces [6]. McDonalds et al. created the three-layer development environment for the CO^{2}P^{3}S project. On that project, the three layers were created as a way to leave some flexibility in the framework; it allows the programmer to create new patterns if needed. We have incorporated that idea into Seeds. Similarly, Aldinucci et al. has provided three layer development environments to explored the use of skeletons as a way to manage non-functional requirements on behalf of the user programmer [11]. Future research will give Seeds non-functional requirement features such as scalability and load-balancing. The same middle layer (the advanced layer) will be used for that purpose. Seeds assumes all the other non-functional requirements, such as security, are done by the Grid middleware. The nested feature is important for any SP framework. Lithium [1], Muskel [2] and other frameworks have implemented the feature [6]. Our work comes closest to the work of Gomez et al. Their pattern operators implement the same concept that set us in the direction to create the pattern operators [12]. Their work is based on workflows and includes other types of operators that are used to manage non-functional concerns. The user programmer, in the Triana framework is given more control over the resources where the program runs. Our work differs from Triana in that we provide the pattern operators as a pure object-oriented framework without the need for XML language constructs, scripts, or a GUI. Also, we have measured the effects of the tool. The use we intend for pattern operators is targeted toward high performance parallel computing in a heterogeneous environment. A similar idea by Gomez et al. that can be useful to port from the problem solving environment (PSE) into SP frameworks is to use pattern operators to add a behavioral pattern (such as check pointing, visualization, and interaction) to a SP parallel application.

5. Conclusion and Future Work

Much of the literature focuses on skeletons, and somewhat on patterns. Our work shows how the number of synchronous patterns can be reduced by using the addition operator. However, it does not show how the synchronous patterns can be scaled automatically for the number of resources. Specifically, the interfaces shown require the user to specify how many processes need to be used to work on the program. A better approach would allow for an automatic scalability of these patterns. Aldinucci has mentioned these goals in his literature. We also believe this goal would better promote patterns for the grid and cloud environments.

We presented an object oriented implementation to provide an adder operator to skeletons/pattern parallel applications. A sample program was shown and its creation was discussed from the user programmer's perspective. Subsequently, the implementation of the pattern adders was presented. The advance user's perspective was discussed, and some notes about the Seeds framework and the expert programmer's perspective was also discussed. The pattern operator can be used to reduce the number of patterns that are needed by the user programmer. We believe that providing a basic set of skeletons/patterns plus useful operators will increase the popularity of this parallel programming model.

6. Acknowledgment

We thank the UMLGraph.org team for providing the tools create class diagrams from source code [13]. The Seeds framework uses packages for SSH access using the Ganymed libraries [14]. Globus [15] access is done through Java Cog [16]. Network overlay organization is done using JXTATM [17], and use of the UPNP communication protocol for routers is also implemented [18]. Thanks to all these software providers.

References

[1] M. Aldinucci, M. Danelutto, and P. Teti, "An advanced environment supporting structured parallel programming in Java," Future Generation Computer Systems, vol. 19, 2003, pp. 611-626.

[2] Marco Aldinucci, Marco Danelutto, and Patrizio Dazzi, "Muskel: an Expandable Skeleton Environment," Scalable Computing: Practice and Experience, 2008, pp. 325-341.

[3] R. Hempel, "The MPI Standard for Message Passing," Proceedings of the nternational Conference and Exhibition on High-Performance Computing and Networking Volume II: Networking and Tools, Springer-Verlag, 1994, pp. 247-252.

[4] L. Dagum and R. Menon, "OpenMP: an industry standard API for shared-memory programming," Computational Science & Engineering, IEEE, vol. 5, 1998, pp. 46-55.

[5] S. MacDonald, K. Tan, J. Schaeffer, and D. Szafron, "Deferring design pattern decisions and automating structural pattern changes using a design-pattern-based programming system," ACM Trans. Program. Lang. Syst., vol. 31, 2009, pp. 1-49.

[6] M. Cole, "Bringing skeletons out of the closet: a pragmatic manifesto for skeletal parallel programming," Parallel Computing, vol. 30, 2004, pp. 389-406.

[7] K. Matsuzaki, H. Iwasaki, K. Emoto, and Z. Hu, "A library of constructive skeletons for sequential style of parallel programming," Proceedings of the 1st international conference on Scalable information systems - InfoScale '06, Hong Kong: 2006, pp. 13-es.

[8] S. Siu, M.D. Simone, D. Goswami, and A. Singh, Design patterns for parallel programming, 1996.

[9] J. Dean and S. Ghemawat, "MapReduce," Communications of the ACM, vol. 51, 2008, p. 107.

[10] T. Mattson, Patterns for parallel programming, Boston [u.a.]: Addison-Wesley, 2007.

[11] M. Aldinucci, M. Danelutto, and P. Kilpatrick, "Autonomic management of non-functional concerns in distributed & parallel application programming," Parallel and Distributed Processing Symposium, International, Los Alamitos, CA, USA: IEEE Computer Society, 2009, pp. 1-12.

[12] M.C. Gomes, O.F. Rana, and J.C. Cunha, "Pattern operators for grid environments," Sci. Program., vol. 11, 2003, pp. 237-261.

[13] "UMLGraph - Declarative Drawing of UML Diagrams," http://www.umlgraph.org/index.html, Mar. 2010.

[14] "Ganymed SSH-2 for Java," http://www.ganymed.ethz.ch/ssh2/, Mar. 2010.

[15] I. Foster, "Globus Toolkit Version 4: Software for Service-Oriented Systems," Network and Parallel Computing, 2005, pp. 2-13.

[16] G. von Laszewski, I. Foster, and J. Gawor, "CoG kits," Proceedings of the ACM 2000 conference on Java Grande - JAVA '00, San Francisco, California, United States: 2000, pp. 97-106.

[17] "jxta: JXTA^{TM} Community Projects," https://jxta.dev.java.net/, Mar. 2010.

[18] "UPNPLib," http://www.sbbi.net/site/upnp/index.html, Mar. 2010.

Compass: a Data Aggregation and Recommendation Technology in the Efficient and Scalable Grid Monitory System (ESGMS)

Weiqing Yang [1, 2], **Xuebin Chi**[1], **Hong Wu**[1], **Siran Ye** [3]
[1]Supercomputing Center, Computer Network Information Center, Chinese Academy of Sciences, Beijing, China
[2]Graduate University of Chinese Academy of Sciences, Beijing, China
[3]Joint Institute of University of Michigan and Shanghai Jiaotong University
E-mail: wqyang@sccas.cn; chi@sccas.cn; wh@sccas.cn; yesiran@163.com

Abstract - *Grid monitoring is a process to collect information regarding the current and past status of large-scale distributed grid computing resources. In the monitoring process, a variety of data packages will be delivered through Internet. When such monitoring system is used by the vast grid computing end users, the potential number of inquiries and the data bandwidth needed to passing the data packages can be significant, which can make the monitoring system intrusive and reduce the remaining communication capacity for the grid computation itself. As the Computer Network Information Center (CNIC) in China, we found that this potential network jamming is an important issue in the daily operation of the grid. This paper presents a "Compass" technology developed in CNIC. We have designed and implemented Compass on our Efficient and Scalable Grid Monitory System (ESGMS), which works on the Scientific Computing Grid (SCGrid), the biggest computing grid in China. Our tests indicate that, compared to the common XML method, the Compass technology can reduce the memory usage by 10%, the computation time by 30%, decrease the response time by a factor of 3, reduce the communication bandwidth usage by a factor of 2, and reduce the number of communication packets by 30%. Compass not only guarantees relatively low performance overhead, it can also provide overall information of the grid and resource allocation recommendations to the end users.*

Keywords: grid monitoring; Compass; Data Aggregation Mechanism; Resource Recommendation Mechanism

1 Introduction

Grid computing aims to integrate distributed, heterogeneous, and possibly miscellaneous computing resources seamlessly for some overall computational tasks [1]. It provides an alternative way to the homogeneous supercomputing. There are varieties of computational problems suitable for grid computing, including: protein folding, climate simulation, earth quake prediction, drug discovery, social and economical simulations. Collectively, the grid can provide huge computational resources. For example, the BOINC (Berkeley Open Infrastructure for Network Computing) project has reached 4.76 Pflops as of March 13, 2010, while the protein folding project: Folding@Home has reached 5 Pflops in March 17, 2009. The operation of the grid is mostly managed by the grid middleware, which helps to parse a computational task into small pieces, and then send these pieces into different resources distributed throughout the grid. While, ideally, everything can be dealt with by the middleware automatically without much user intervention, in reality however, it is highly desirable to have a human monitory system which can track the current status of the grid, the usage loads of different parts of the grid, the communication/traffic information, the dispatching pattern of a given job, and recommends the resources to users. With such monitory system, the user can make human decisions for how and where to launch their jobs. To provide these services, the monitory system needs to collect vast amount of data for fault detection, performance analysis, performance tuning, performance prediction, and scheduling [3]. Overall, grid monitoring is a critical part of the grid computing. It can also be viewed as a part of the grid middleware.

The developments of grid system prototypes and grid monitoring technologies are being carried out in a number of countries and regions, including the United States, European Union, China, Japan, and Korea, etc. In China, the grid has aroused the interests of many researchers in information technology and applied sciences [12]. China is experiencing a rapid growth in internet bandwidth and computing resources. The desire for cooperation and volunteering resources is abundant, and there are national initiatives for large scale grid computing projects. A test-bed for the grid technologies, the Scientific Computing Grid (SCGrid), is being developed in Chinese Academy of Science under the supports of the National High-tech Research and Development Program (the 863 program) and Informatization Construction Project of Chinese Academy of Sciences during the 11th Five-Year Plan Period. SCGrid consists of thirteen high performance computing centers across the country. Applications running on the grid includes: protein folding, earth quake simulation and financial simulations etc. The goal of SCGrid is to promote the development and application of high performance computers and to develop grid technologies that enable resource sharing and cooperative work in the Internet environment. The Efficient and Scalable Grid Monitory System (ESGMS) is developed in our institute as a monitoring system for SCGrid.

The purpose of ESGMS is to satisfy the requirements of all the grid users, including: Grid operators, site administrators, Virtual Organization (VO) managers and Grid end users. It used GMA (Grid Monitoring Architecture) monitoring architecture [13], the most important feature of which is the separation of the discovery and retrieval operations. We have adopted the Web Service to implement the GMA in our ESGMS. However, there are particular issues need to be addressed in using the Web Services. Both GMA and Web service can deal with loose communication effectively for the heterogeneous systems. However, using Web Service, the system reporting data is often complex and bulky [10][11]. For example, Web service uses XML for data representation, which significantly increases the data volume and transmission frequency. In real operation, due to the large number of users and inquiries, this could cause communication traffic jam and hamper the non-intrusiveness of ESGMS. Thus, a light weight communication technology is needed. In this paper, we present the "Compass" technology for this aim, which is a central part of ESGMS. Compass reduces the package size and frequency by onsite data aggregation guided by past statistics.

There are many ways to reduce the communication burden of the monitoring system. Data Compression Technology--compressing XML can be used to reduce the XML data volume before transmission, but the compression and decompression take time. Optimizing XML parser is another method, but its RPC technology increases waiting time for a remote call. As we will show, Compass technology avoids these pitfalls, and it is based on intelligent data aggregation and pre-processing, and is more efficient than these straight forward methods.

In addition to provide the information for the overall status of the grid, the monitoring system is also crucial for the following activities (cases): scheduling, performance analysis and optimization of distributed tasks or individual applications. Although some monitoring systems have proposed specific API to those cases, their overhead may cause performance problems for grid monitoring system. To address this problem, Compass also developed a resource recommendation mechanism.

Thus, Compass has two aspects: Data Aggregation Mechanism (DAM) and Resource Recommendation Mechanism (RRM). DAM focuses on data aggregation to yield smaller packages and less number of packages when the number of sensory data inquiry increases sharply. The DAM is an intelligent based agent. It guesses what do the average users want from past statistical data of the users' sensory inquiries. Based on such predictions, a local DAM agent will pre-processes the local raw data, and prepare a local database for different kind of summary and possible inquiries. Such aggregated and summarized data distributed at different sites can be used to fill in the actual inquiry when requested, thus provide much smaller package size, and faster response time.

As implemented today, the ESGMS can aggregate data either by extracting or calculating the most concerned data of users from the Web service provided XML, or directly sensor the resource layer. RRM is in charge of recommending suitable resources to users based on the information from DAM through a recommendation algorithm.

The remainder of this paper is organized as follows. First, in section II, we briefly describe the overall architecture of ESGMS and the component design of Compass. Section III illustrates the data preparation process in Compass. Section IV introduces the details of the data aggregation mechanism (DAM), including the design and implementation of DAM and the time complexity analysis of aggregation. Section V introduces RRM recommendation algorithm. In Section VI, we present our evaluation of Compass and analyze the testing results. At last, we summarize the work and discuss future research directions.

2 System overview

Figure 1 shows the architecture of ESGMS. ESGMS includes three layers: Resource Layer, Monitoring Service Layer and Presentation Layer. Note that, these three layers are different from the usual three layers structure of the grid computing itself, which are: user, middleware and resources. Resource Layer of ESGMS consists of a variety resources of grid system, such as Ganglia [5][6], PBS, Middleware, etc. Monitoring Service Layer provides public monitoring services for resource information retrieving, aggregating, processing and delivering. Presentation Layer uses data or API both offered by Monitoring Service Layer for web presentation, scheduling, data replication, accounting, performance analysis and so on. Some snapshots of the presentation layer are shown in Fig 2.

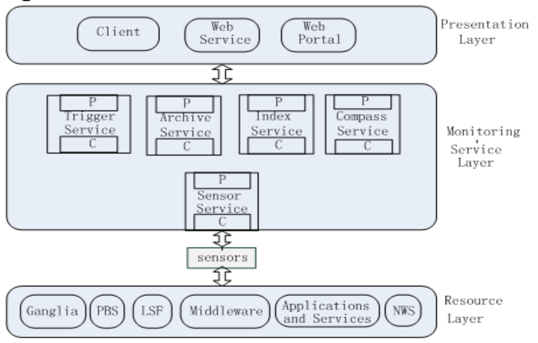


Figure 1 · The architecture of ESGMS

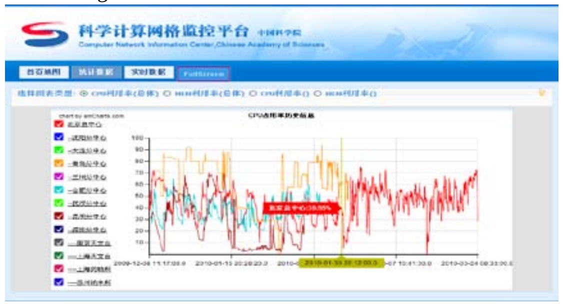

科学计算网格监控平台
Computer Network Information Center,Chinese Academy of Sciences

Fig.2.1 The situation of CPU usage history (different lines indicate different centers)

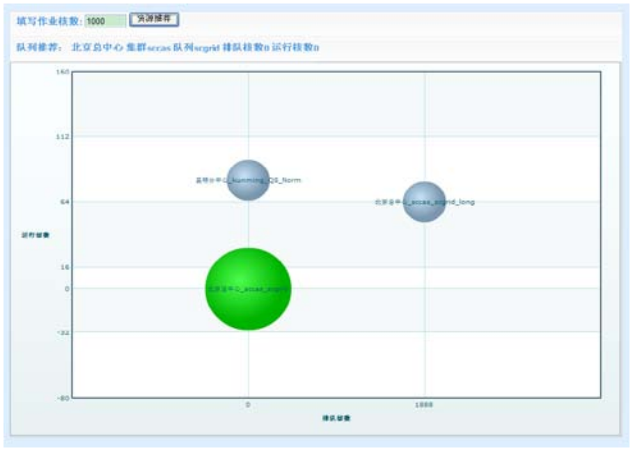


Fig.2.2 The recommendation of Compass (the big green ball is the recommended object)

In the Monitoring Service Layer, there are five subservices. Trigger Service is used to report alarm messages to system manager by e-mail or other ways. Archive Service is in charge of collecting historic data. Index Service is mainly developed to locate, name and describe the grid data with structured characteristics [7]. Sensor service is to manage sensors' working in the lower layer, which helps to build a hierarchical architecture to achieve better scalability. Finally, the Compass service is used for releasing aggregated information and recommending suitable resources to users. Note that, in our current design, the Compass Service can be switched ON/OFF in the Monitoring Service Layer. This provides us a way to test our Compass system compared to the conventional direct XML based system.

ESGMS also adopts many other technologies such as Ganglia, LSF and PBS for sensors, XML/AL for data representation, Web Service for data transport, Java Servlet for MSU, Ajax for data display request [8], Amcharts and FusionCharts for chart generation [9]. In the rest of the paper, we will mainly focus on the Compass Service. Figure 3 shows the component design of Compass, and its relationship to other services and layers in ESGMS.

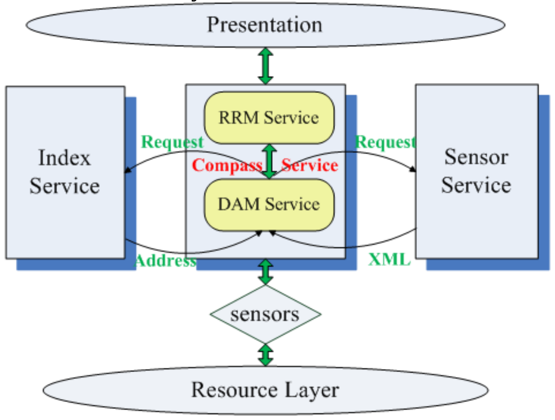


Figure 3 · The component design of Compass

3 Data preparation

ESGMS is targeted at satisfying the requirements from the following main categories of users: Grid managers, site administrators, Virtual Organization (VO) managers and Grid users. Grid managers have the highest authority, they can monitor and manage all sites. Site administrator and Virtual Organization (VO) can monitor and manage the site situation or the VO situation respectively. Grid users can browse the overall grid information, including CPU usage, Memory usage, average load of all sites. The end users can only see their own job status, but cannot see other users' job situation. All of our grid users access ESGMS through web interface. Before we designed the Compass service, we had made a survey on what are the users' most concerned data through a users' login ESGMS questionnaire. Through the collecting and analyzing the results of these questionnaires, we found the following:

Grid users (the fourth class of users) play an important role (99%) in contributing to the suddenly sharp increase of data inquiries. On the other hand, they only care about some specific information of their own jobs running on the grid, such as: whether the operation status is running or hang, the total operating time, the queue situation at each site， which queue in one site is idle or has high CPU usage and Memory usage, the node with the least number of waiting jobs, etc. After the users understand the current situations, they can submit their jobs to the suitable resource of grids. However, they do not care about the situations of all nodes in each site. A naïve monitoring system using all the data provided by the Web service XML for all the nodes will give too much unneeded information, and overwhelm the network. Based on this understanding, we have designed a new ESGMS with Compass. Compass uses DAM to aggregate data from XML which are already generated by Sensor Service and the data directly obtained by sensors. These are primitive detail raw data with high data volume. Compass then extract or process the data onsite, and only transmit the user interested information through the net. The results in smaller data packages and less number of packages, but at the mean time, still satisfy the need for various users of ESGMS.

4 Data aggregation mechanism (DAM)

Compass's DAM continuously computes summaries of the data filtered from XML already generated by Sensor Service or offered directly by sensors, and it asks Index Service for resource addresses. DAM is controlled by SQL queries, and can be viewed as a type of data mining process like a virtual database which does not reside on a centralized server and not support atomic transactions. DAM is also intended as a summarizing mechanism which uses aggregates that summarize the overall state for the grid system as a whole, and also for the local domain for which a DAM agent is responsible for. DAM can be accessed and manipulated by using database integration mechanisms like ODBC and JDBC standard database programming tools.

DAM must limit (set an upper bound) the rate of information flow at each participating node, so that even under worst-case scenario, the traffic will be independent of system size. For example, DAM can count the number of nodes having some given property, but not to concatenate their names into a list, since that list could be of unbounded size. Each aggregate in DAM is restricted to some domains, within which it is computed on behalf of and visible to all nodes. Only aggregates with high global value should have global scope. As grid information changes, DAM will automatically and rapidly re-compute the associated aggregates and report the changes to Recommend Service which will recommend new suitable resources (services) to users.

4.1 Design and implementation of DAM

In order to support self-management and self-monitoring better, ESGMS system uses the distributed multi-center structure and adopts center resource autonomy ideology: setting up local resource monitoring system, then integrating the information of various local resources, and providing a unified query interface. Therefore, the whole grid system is divided into two layers physically: the global monitoring and the local monitoring. This structure can prevent single point of failure, facilitate system software upgrade, and improve the system scalability. Thus the ESGMS is a hierarchical tree structure, the overall framework is as follows:

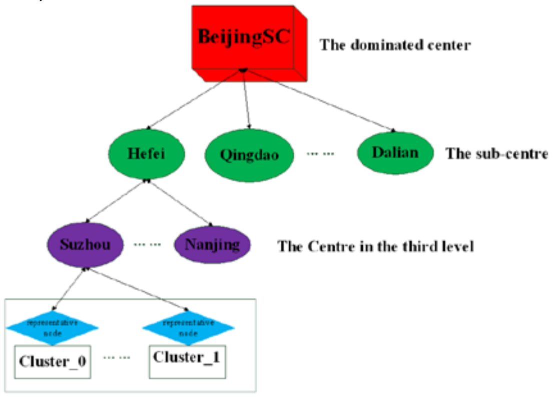


Figure.4 The hierarchical architecture of ESGMS

Figure.4 shows the architecture of ESGMS, which adopts hierarchical architecture style viewed as a three-tier structure and has a corresponding relationship with actual physical resources and connections. In the Figure.4, the red frame indicates Beijing Supercomputing Center (BeijingSC) of the first level of ESGMS three-tier structure. The cyan ellipses mean sub-centers in the ESGMS second level, such as Dalian Center, Qingdao Center, etc. The violet ellipses represent centers in the third level, such as Suzhou Center, Nanjing Center, etc. Authority becomes smaller and smaller form the first level to the third. Each center has its own clusters with one representative node to aggregation. Compass agent will be deployed in the representative node of each cluster, and aggregates not only the information of each cluster but also the information of its child agent. Compass agent on each cluster offer external API. The tree structure of the ESGMS and the Compass is in line with the actual physical network topology and connection of the grid. This would help to reach a balanced and efficient Compass system. If the representative node in a cluster deployed with Compass agent fails or becomes unsuitable, Compass will automatically select another node to take its place.

Each representative node in a cluster runs a Compass agent. System administrator initializes these agents with their unique names which are local domain identifiers within their parent domain. Those names are strings consisting of their path of domain identifiers from the root, separated by dots (e.g., “BeijingSC.ShanghaiSC.Cluster0 "). Compass hierarchy is implicitly specified when the agents are initialized. For example, the ‘BeijingSC.Dalian.Cluster0’ agent creates the ‘BeijingSC’ domain and the ‘BeijingSC.Dalian’ domain if they did not exist already. Thus the Compass hierarchy is formed in a decentralized manner.

Associated with each Compass agent is an Attribute List (AL) which contains the information associated with the domain. AL attributes are not directly writable, but generated by so-called Computing Function (CF), through which one could query the states of resources. Each Compass agent has a set of CFs calculating attributes for the center's ALs. CF is an SQL program, which takes a list of the ALs of the center’s child center and produces a summary of their attributes.

If one thinks of Compass as a form of decentralized hierarchical database, AL will be a table (a relation) for each center, with a row for each child center. Each column in a leaf node is a value extracted from the corresponding object on that node. Each column in an internal center is a value computed by CF to summarize its children. These columns might be very different from those of the children centers. For example, the child centers might report lowest loads, biggest CPU usage, etc. In comparison, their father domain could have one column giving the mean load on its children, another counting the biggest CPU usage based on its children, and a third listing the three child nodes with the strongest matches.

AL is relatively small objects, which is bounded in size (about a few hundred or even thousand bytes). ALs export information about resource, such as a time stamp, biggest CPU usage, average load and so on. One might design a CF to count all nodes that match some predicates, yet might not use CF to make a list of all such nodes. In fact, by traversing the Compass hierarchy it is easy to enumerate the nodes that contribute to a counting calculation. We will analyze the time complexity of traversing DAM Tree in the next section.

CFs are programmable. The code of CFs is embedded in the so-called Computing Function Letters (CFLs), which are signed and time stamped certificates that are installed as attributes inside ALs. CFLs also specify two other important questions: what information users want to retrieve at each participating node, and how to compute this information in the Compass hierarchy.

Table1. Computing Function Interface (CFI)

Method	**Description**
FindContacts(time, center)	find Compass agents in the given center
SetContacts(IP)	specify addresses of initial agents to connect to
GetAttributes(center, new values)	report updates to attributes of center
GetChildren(center, new values)	report updates to domain membership
SetAttribute(attribute, new value)	update the given attribute

Users can invoke Compass interfaces through calls to CFs (see Table 1). Besides CFs native interface, Compass has an SQL interface that allows users to view each center as a relational database table, with a row for each child zone and a column for each attribute. The programmer can then simply invoke SQL operators to retrieve data from the tables. Using selection, join, and union operations, the programmer can create new views of the DAM data. An ODBC driver is available for this SQL interface, so that many existing database tools can be used in Compass directly, and many databases can import data from Compass. SQL does not support instant notifications of attribute changes, so that applications that need such notifications would need to obtain them using the native interface. Compass agents also act as web services, hence information can be browsed using any standard web browser instead of going through the CFI.

4.2 Time complexity analysis of DAM aggregation

As mentioned before, the ESGMS architecture tree is based on the actual physical distribution of Grid resource, which makes it scalable for large centers/layers. In the DAM computing process, we need to enquire nodes from the root of ESGMS Tree to leaf-node recursively. It means that we appoint some search qualifications and reference values to find the nodes matching our search conditions by traversing ESGMS Tree from its root. Traversing step number is equivalent to the depth of ESGMS Tree (log (N)). So the time complexity of traversing ESGMS Tree is O(1og (N)), the average time complexity is also O(1og (N)).

Above all, the hierarchical structure of ESGMS Tree has reduced the time of CFs inquiring nodes' ALs, and the time complexity is just log (N).

5 Resource recommendation mechanism (RRM)

Based on DAM, we can design CFs or use SQL statements to get desired information to users. For example, we can get a node's URL which has the biggest CPU usage in a domain (even the top several nodes), or an average Memory usage of a domain, as follows:

```
        SELECT
            CPU
         WHERE CPU= (SELECT MAX (CPU));
          SELECT
            AVE (MEM) AS MEM
          WHERE ZONENAME = DEEPCOMP 7000;
```

The computing results of CFs or SQL statements can serve as inputs of RRM. RRM recommends resources (services) to grid users based on the following algorithm. Here we define nodes in Grid system as the resources to be recommended.

We use PPC (Potential Processing Capability) to indicate the potential processing capability of node. PPC is affected by two factors: Available Resource Factor (ARF) and Waiting Tasks Factor (WTF).

ARF indicates the processing capability of the node. The value of ARF is related to the Available CPU Usage (ACU) and Available Memory Usage (AMU) of node.

WTF indicates the degree of tasks assignment on the node. We define the value of WTF as the number of tasks waiting (NTW) on the node.

ARF have a positive effect on PPC, while WTF has a negative effect on PPC. To combine their effects, PPC for a given node is calculated as in the following formula:

$$PPC = \frac{\frac{ACU}{\sum ACU} * \frac{AMU}{\sum AMU}}{\frac{NTW}{\sum NTW}} \quad (1)$$

The summation goes for different nodes in a domain. The recommendation mechanism works as follows: Supposing there is a resource set NODE = $\{N0, N1, \ldots, Nn\}$; ACU[i], AMU[i], NTW[i] and PPC[i] respectively representing the value of ACU, AMU, NTW and PPC of node Ni. The best node recommended to the users satisfies:

$$PPC[i] = \max\{PPC[j]\}\left(j = 0,1,\ldots,n-1\right) \quad (2.1)$$

$$ACU[i] > 0 \quad (2.2)$$

$$AMU[i] > 0 \quad (2.3)$$

Formula (2.2) and Formula (2.3) indicate that Compass cannot recommend the unavailable resource to users.

However, in order to compare the PPC of two nodes i, j, we don't really need to compute PPC (which involves summation). Instead, we can judge that from the following formula.

Firstly, we can define the symbol "?" as the meaning of "comparing different nodes for PPC value", and Ni, Nj are the nodes in NODE set. PPC values of Ni and Nj are respectively as follows:

$$PPC[i] = \frac{\frac{ACU[i]}{\sum ACU} * \frac{AMU[i]}{\sum AMU}}{\frac{NTW[i]}{\sum NTW}}$$

$$PPC[j] = \frac{\frac{ACU[j]}{\sum ACU} * \frac{AMU[j]}{\sum AMU}}{\frac{NTW[j]}{\sum NTW}}$$

Thus formula (2.1) (PPC[i] ? PPC[j]) is also written as:

$$\frac{\frac{ACU[i]}{\sum ACU} * \frac{AMU[i]}{\sum AMU}}{\frac{NTW[i]}{\sum NTW}} ? \frac{\frac{ACU[j]}{\sum ACU} * \frac{AMU[j]}{\sum AMU}}{\frac{NTW[j]}{\sum NTW}} \quad (3)$$

Since ΣACU, ΣAMU and ΣNTW are constants in this recommendation, there is no need to divide by ΣACU, ΣAMU and ΣNTW. Formula (3) can be simplified as:

$$\frac{ACU[i]*AMU[i]}{NTW[i]} ? \frac{ACU[j]*AMU[j]}{NTW[j]} \quad (4)$$

It can be further rewritten as:

$$ACU[i]*AMU[i]*NTW[j] ? ACU[j]*AMU[j]*NTW[i] \quad (5)$$

Therefore, comparing two nodes for PPC value can directly use formula (5), there is no need for perform the summation. Algorithm 1 shows the code of RRM's recommendation algorithm.

```
Algorithm 1 Recommendation()
Zone_Num = Count( Zone_Set )
for all i such that 0 ≤ i < Zone_Num do
  DAM(Node_ACU > 0, Node_AMU > 0, PPC[two Node_Max])
  Queue ( result_queue, PPC[two Node_Max] )
end for
if QueueExist == true then
   return Queue
else
   return no resource
end if
```

6 Test and evaluation

In order for Compass technology to be widely used, it must first meet the prerequisites of having efficiency and low performance overhead. To quantify this, we have carried out a series of experiments to evaluate the Compass performance implemented in ESGMS. We have been focused on the execution overheads and the response time. The response time denotes the average amount of time required for a recommendation from being sent till response received.

6.1 Experiment environment

At this stage, the experiments were done in the Supercomputing Center of CAS. In the future, we will perform a series of experiments in more centers. The experiments were run on the SCEYE Test bed at Supercomputing Center of CAS, including three computers with names PC{ 0, 1, 2 }and a high performance computer DeepComp7000, which has ranked No. 19 in the TOP500 Supercomputer list in November, 2008. DeepComp7000 has two clusters: IBM HS21 Node cluster and IBM X3950 Node cluster. The machines are connected within the 100M Ethernet. PC0, PC1 and PC2 are equipped with one 2590MHz Intel Core Duo CPU and 2G main memory. PC0 runs Windows XP Profession Service Pack3 as its operating system. PC1 and PC2 run Linux kernel 2.6.18. Each of PC0-2 and DeepComp7000 has a corresponding part of ESGMS deployment. PC0 is used to Web display. PC1 is the representative (aggregating) node of the IBM HS21 Node cluster, PC2 is the representative (aggregating) node of the IBM X3950 Node cluster. PC1-2 run Compass agent. The monitoring system ESGMS retrieve resource information from DeepComp7000 periodically. The size of XML is about 1.5M, but AL is just about 550 bytes.

This test was done over a 12 hour period on ESGMS system. During that test period, there are 4,000 jobs running on the DeepComp7000 machine.

6.2 Experiment strategies and results

6.2.1 Overhead and Responsiveness

Table 2 shows local per-node monitoring overhead in ESGMS with and without Compass. As mentioned above, the Compass system can be turned off. Without Compass system, the communication is done with the conventional XML method.

Table 2, Compass overheads

System	CPU(%)	PhyMem(MB)	VirMem(MB)
ESGMS without Compass	<0.56	8.0	10.02
ESGMS with Compass	<0.405	7.2	9.01

From the experimental results, we can see that, the performance overhead (both for CPU and memory) of ESGMS with Compass is lower than ESGMS without Compass. The reduction is about 30% in CPU and 10% in memory.

Table 3 compares Compass-based recommendation method with XML-based recommendation method for the response time, net bandwidth usage, and packet numbers.

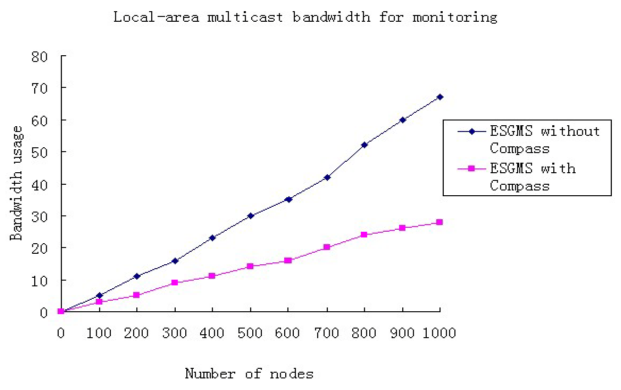
Table 3 Compass response time, bandwidth usage, and packet numbers.

Mechanism	Respons e time(s)	Net bandwidth Usage (Kbits/s)	Packets (number/s)
Compass	0.5	30	550
Common method (XML)	1.8	68	850

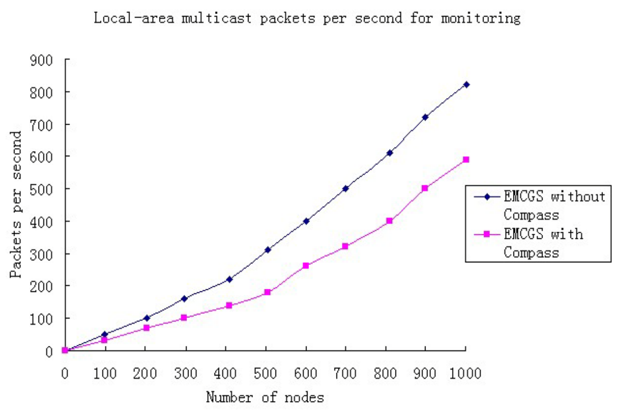
The response time in table.3 denotes the average amount of time required for a recommendation from being sent till response received. Net bandwidth usage indicates Network bandwidth consumed for local monitoring. Packets indicate the number of packet casted per second. From the experimental results, we can see, the response time of Compass is a factor of 3 smaller than Common method based on XML. The Network bandwidth consumed for local monitoring is factor of 2 smaller, and the number of packets casted per second has decreased by 40%. These are significant improvements. As we discussed before, the current test is only for a limited grid system. We expect an even bigger saving for the real larger grid systems as the information increases dramatically in the conventional XML method, while it is bounded in our Compass system, as it will be shown in the following scalability test.

6.2.2 Scalability

In the following experiments, we measure the scalability of ESGMS with Compass system and ESGMS with the Common method (XML) as we scale the number of nodes within DeepComp7000.



(a) Local-area multicast bandwidth usage for monitoring



(b) Local-area multicast packets per second for monitoring

Figure 5 · Scalability as a function of cluster size

In Fig. 5a and b, we quantify the scalability of ESGMS with Compass running on DeepComp7000. As a direct consequence of using native IP multicast, we observe a linear scaling of ESGMS with Compass in local-area bandwidth usage as a function of DeepComp7000 size. Although ESGMS without Common method (XML) is also linear, the slope of it is bigger than ESGMS with Compass. We also observe a linear scaling in packet rates of both ESGMS with Compass and with Common method (XML), but the slope of former is less than the later, again due to our use of native IP multicast as opposed to point-to-point connections in XML. The reduction in the bandwidth usage and the number of packets can be at least partially attributed to ESGMS's use of thresholds.

In conclusion, Compass maintains excellent responsiveness even as the system becomes very large, and even if it exhibits significant dynamic. The work loads associated with the technology are small and bounded, both for the message rates seen by participating machines and loads imposed on communication links. Compass also has a small "footprint" in the sense of computational and storage overheads. ESGMS with Compass has better performance than ESGMS without Compass. The improvements at different aspects range from 30% to a factor of 3.

7 Conclusion

In this paper, we have presented a Compass technology based on DAM and RRM. We have designed and implemented Compass technology in ESGMS. We show that significant improvement can be achieved in terms of reducing the computation and communication overheads and memory footsteps of the monitoring system.

Our tests show that, compared to the common XML method, the Compass technology can reduce the memory usage by 10%, the computation time by 30%, increase the response time by a factor of 3, reduce the communication bandwidth usage by a factor of 2, and reduce the number of communication packets by 30%. These are significant improvements. Our test also shows that the Compass has a linear scaling to the size of the system. The design and architecture of the Compass is discussed in the paper. The algorithm used for the recommendation system is presented. While we are still at the stage of further improving the Compass system, we believe this system will play a very important role in our ESGMS. It satisfies the needs for a good monitoring system: low overhead, low intrusiveness, high scalability and efficiency. These results show that we can develop a very efficient and flexible grid monitoring system based on the Compass technology. In the future, we will perform a series of larger scale experiments including more centers to evaluate the Compass performance. Furthermore, we intend to explore new recommendation methodologies based on some forecast methods, such as [4], to enhance recommendation capability of Compass including accuracy and non-intrusion.

ACKNOWLEDGMENT

We wish to thank Astrolabe system [2] for giving us some inspiration when we developed ESGMS. This work was supported by grants from Informatization Construction Project of Chinese Academy of Sciences during the 11th Five-Year Plan Period No.INFO-115-B01, the 863 program No.2006AA01A117, and the 863 program No.2006AA01A116.

REFERENCES

[1] Jennifer M. Schopf, Ioan Raicu, Laura Pearlman, Neil Miller, Carl Kesselman, Ian Foster, Mike D'Arcy, Monitoring and Discovery in a Web Services Framework: Functionality and Performance of Globus Toolkit MDS4, May 2006.

[2] ROBBERT VAN RENESSE and KENNETH P. BIRMAN and WERNER VOGELS, "Astrolabe: A Robust and Scalable Technology for distributed System Monitoring, Management, and Data Ming," ACM Journal Name, Vol. V, No. N, Month 20YY, Pages 1-42.

[3] N. Podhorszki, Z. Balaton, G. Gomb´as, Monitoring message passing parallel applications in the grid with GRMand mercury monitor, in: Proceedings of the Second European Across Grids Conference, Nicosia, Cyprus, 2004..

[4] Dengpan Yin, Esma Yildirim, Tevfik kosar. A Data Throughput Prediction and Optimization Service for Widely Distributed many-Task Computing. *MTAGS'09* November 16th, 2009, Portland, Oregon, USA.

[5] FD Sacerdoti, MJ Katz, ML Massie, DE Culler, Wide Area Cluster Monitoring with Ganglia, In the Proceedings of the IEEE Cluster 2003 Conference, pp.289-298, 2003.

[6] Matthew L. Massie, Brent N. Chun, David E. Culler, The Ganglia distributed monitoring system: Design, Implementation, and Experience, Parallel Computing, Vol. 30, pp. 817-840,2004.

[7] Xuehai Zhang, Jeffrey Freschl, Jennifer M.Schopf, A Performance Study of Monitoring and Information Services forDistributed Systems, Proceedings of HPDC, vol. 30, August, 2003.

[8] Open Source Application Foundation, Survey ofAJAX/javaScript libraries (24 libraries surveyed), http://wiki.osafoundtion.org/bin/view/Projects/AjaxLibraries, 2007.

[9] Amcharts Foundation, Amcharts, the JavaScript toolkit, http://www.amcharts.com.

[10] World Wide Web Consortium, http://www.w3.org.

[11] Fumio Machida, Masahiro Kawato, Yoshiharu Maeno, Guatantee of Freshness in Resource Information Cache on WSPE: Web Service Polling Engine, Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid,Vol.1, pp.131-134, June 2006.

[12] Qian Depei. CNGrid: A test-bed for Grid Technologies in China. Proceeding of the 10th IEEE International Workshop on Future trends of Distributed of Distributed Computing Systems (FTDCS'04).

[13] B. Tierney, R. Aydt, D. Gunter,W. Smith, M. Swany, V. Taylor and R. Wolski, A Grid Monitoring Architecture, Technical report, Global Grid Forum, January 2002.

A New Technique to Improve the Makespan in Computational Grids

Ass. Prof. Nahed M. El Desouky[1] , Ass. Dr. Bayoumi. M. Ali Hassan[2],and Afaf Abd El-kader Abd El-hafiz[1]
[1]Department of Mathematics, Faculty of Science , Al-Azhar University(girls), Cairo, Egypt.
{nahedmgd@yahoo.com, afafoafaf@yahoo.com}
[2] DS Dep., Fci-Cu, Cairo University ,Cairo , Egypt.
{bayoumi2000@hotmail.com}

Abstract - *Divisible workload scheduling on distributed systems has been one of the interesting research problems over the last few years. Most of scheduling algorithms are static, that is, they assume that grid resources such as CPU power and network bandwidth are constant. The most valuable static scheduling algorithm for scheduling divisible loads on distributed systems is the UMR (uniform multi-round) algorithm. A dynamic extension to the UMR is proposed by Elnaffar and Nguyen in [1]. They use the Mixed Tendency Based prediction strategy to predict the CPU utilization. This paper proposes a scheduling technique based on Average Prediction of Utilization (APU) for predicting the CPU utilization. The proposed technique is compared with a scheduling technique based on the Mixed Tendency Based prediction technique. The results show that the APU method reduces the makespan by 17.5% to 13% depends on the length of the chunk. Also, with same workload and different number of workers the makespan is improved by values varies from 16.5% to 2% .We present simulation results to quantify the benefit of our technique.*

Keywords: Divisible workloads, Grid computing, CPU utilization predction.

1 Introduction

Some applications require large amounts of computer resources. These applications consist of many independent computational tasks and arise in many fields of science and engineering [2]. The problem of scheduling these applications on a network of computing resources is addressed in [2] and it is studied for two different application models: fixed-sized tasks and divisible workloads. In the first model of applications, workload consists of a number of tasks whose size is pre-determined. In the second model, the divisible workload is continuous and the scheduler can partition the workload in arbitrary different tasks (chunks). In this paper we focus on the divisible workload problem i.e. the problem of how to divide an application's workload into many parts and assign them to computers (workers) in a distributed environment (grid) so that the execution time is minimum[2].

Most of the scheduling algorithms assume that computational resources are dedicated, which makes these algorithms not practical for distributed systems as grids. In grids, computational resources are expected to serve local tasks in addition to the Grid tasks. A shortcoming of these algorithms is that they do not take the dynamicity of Grids into account. The UMR (uniform multi-round) algorithm is the most valuable static algorithm that is used to solve the divisible load problem. By this algorithm, the workload is divided into multiple rounds where each round is divided into identical chunks which are dispatched to computer resources. The workload is submitted to the scheduler for processing. In order for the scheduler to decide how to divide up the workload and distribute the chunks on the workers, it needs to know the available computational power (actual CPU speed) of each worker. Therefore, the scheduler queries a prediction component that performs short-term prediction by collecting and analyzing a series of CPU utilization values [1]. An improvement to the UMR algorithm is introduced in [1] which makes it dynamic. They use the Tendency Based Prediction method which is one-step-ahead that takes into consideration the ascending and descending behavior of the CPU load of a worker. By predicting the CPU load (or utilization), the speed of the worker at that time can be computed and feed such information to the UMR algorithm in order to make better scheduling.

The workers of Grid may be tree-shaped, star-shaped or any other shape and they may be independent at all. The star-shaped network consists of a master and a number of workers. There is a communication link from the master to each worker. In tree-shaped network, the tree is rooted at the master that has a link to each one of its children. Each one of the master's children may also have children to which they can delegate work. Scheduling divisible loads in a single-level tree network is considered in [3]. A number of significant results on load sequencing and processor-link arrangement is proved for this case. Scheduling Divisible Loads on Star and Tree Networks is studied in [4] which gives many results and open problems for this case. Here, we will concentrate on the star-shaped network that consists of a master and a number of workers. The master uses its network connection in a sequential fashion. It does not send chunks to workers simultaneously. This assumption is justified either by the master's implementation, or by the properties of the network links [1].

This work is organized as follows. The next section describes the application and platform models of the network considered in this work. The UMR algorithm is discussed in section 3. Section 4 briefly describes the prediction method used to predict the CPU utilization method and the algorithm that based on it. Section 5 gives the proposed technique APU. Section 6 gives the experimental results. Finally, we conclude and sketch future work in section 7.

1.1 Instructions for authors

An electronic copy of your *full camera-ready paper* must be uploaded (in PDF format) to Publication Web site before the announced deadline. Please follow the submission instructions shown on the web site. The URL to the website is included in the notification of acceptance that has been emailed to you by Prof. Arabnia.

2 Models

The following Grid model is considered in this paper:

2.1 Applications

This work considers applications that consists of a workload, W_{total} that is continuously divisible. The scheduler determines the size of a chunk (the portion of workload that is given to a worker). Only the transfer of application data input will be considered.

2.2 Computing Platform

The model considered here is a star-shaped master/worker model with N workers. Chunks are sent out to workers over a network by the master which uses its network connection in a sequential fashion. It does not send chunks to workers simultaneously. This is a common assumption and is justified either by the master's implementation, or by the properties of the network links[1]. The speeds of network communications to each worker need not be identical. Therefore, the platform topology consists of network links with various characteristics to the set of heterogeneous processors, as depicted in Figure 1.

It is assumed that the workers can receive data from the network and perform the computation simultaneously. To formalize this model, consider a portion of the total workload $chunk_i \leq W_{total}$ which is to be processed on worker w_i, $1 \leq i \leq N$. The time required to worker w_i to perform the computation, T_{compi} is modeled as

$$Tcomp_i = cLat_i + \frac{chunk_i}{S_i}, \qquad (1)$$

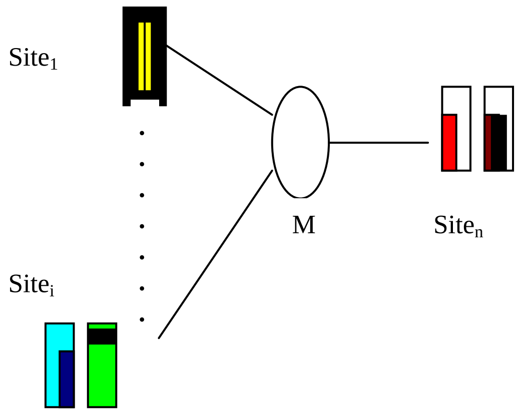


Figure 1 Computing Platform Model

Where $cLat_i$ is a fixed overhead, in seconds, for starting a computation (e.g. for starting a remote process), and Si is the computational speed of the worker in units of workload performed per second. Computation, including the $cLat_i$ overhead, can be overlapped with communication. We model the time spent for the master to send chunk units of workload to worker w_i, T_{commi}, as:

$$Tcomm_i = nLat_i + \frac{chunk}{B_i} + tLat_i, \qquad (2)$$

where $nLat_i$ is the overhead, in seconds, incurred by the master to initiate a data transfer to worker w_i (e.g. pre-process application input data and/or initiate a TCP connection); B_i is the data transfer rate to worker w_i, in units of workload per second; $tLat_i$ is the time interval between when the master finishes pushing data on the network to worker w_i and the time when worker w_i receives the last byte of data. It is assumed that the $nLat_i$ + $chunk/B_i$ portion of the transfer is not overlappable with other data transfer. . All section headings including the subsection headings should be flushed left.

3 The UMR Algorithm

The UMR algorithm [1] dispatches chunks of the workload in rounds. We have

$$W_{total} = \sum_{i=1}^{M} round_j \qquad (3)$$

$$round_j = \sum_{i=1}^{N} chunk_{j,i} \qquad (4)$$

Where

- *roundj*: amount of workload the master delivers during round *j*,

- $chunk_{j,i}$: the fraction of the total workload, *Wtotal* , that the master delivers to worker *i* in round *j* ($1 \le i \le N$; $1 \le j \le M$).

- *M* is the number of rounds required to dispatch all chunks to workers. The computation of M will be given shortly

The UMR algorithm splits the workload into chunks in such a way that each worker in *roundj* finishes its computation in a constant time, $const_j$. That is

$$cLat_i + \frac{chunk_{j,i}}{ES_i} = const_j (\forall i = 1..N) \qquad (5)$$

By combining the last two equations, we obtain a simple induction relation on the chunk sizes

$$chunk_{j,i} = \alpha_i \times round_j + \beta_i \qquad (6)$$

Such that

$$\alpha_i = \frac{ES_i}{\sum_{k=1}^{N} ES_k} \qquad (7)$$

$$\beta_i = \frac{ES_i}{\sum_{k=1}^{N} ES_k} \sum_{k=1}^{N} (ES_k \times cLat_k) - ES_i \times cLat_i \qquad (8)$$

Complete details on these derivations are provided in technical report [5].
Figure. 2 shows how UMR algorithm dispatches chunks of workloads in multiple rounds.

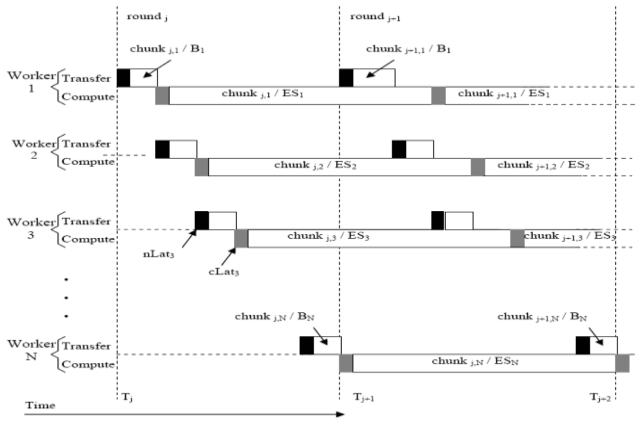


Figure 2 UMR dispatches workload chunks in rounds

In order to achieve overlapping between communication and computation, once the last worker *N* starts processing its chunk received in *roundj*, the master starts sending *N* chunks for $round_{j+1}$ ($chunk_{j+1,1}$, $chunk_{j+1,2}$, ..., $chunk_{j+1,N}$) to *N* workers. To maximize bandwidth utilization, the master must finish sending all chunks of $round_{j+1}$ to all workers before worker *N* finishes its computation for *roundj*

4 Scheduling based on Tendency Based Prediction method

In grids, workers are responsible for executing their local tasks and, if they become underutilized, they can handle incoming grid tasks. The priority is given to local tasks. Consequently, and depending on the local load, we cannot always assume the availability of the full processing speed, *S*, to Grid tasks. Based on the measured CPU utilization, the *ActualSpeed* that is available for Grid tasks can be computed as follows

$$ActualSpeed = S * (100\% - Utilization)$$

Therefore, if we predict the *Utilization* of a worker, we can compute and send the anticipated processing speed (*ES*) to the scheduling algorithm. The time series prediction approach proposed in [6, 7] is used which has been empirically effective in predicting the CPU load and utilization. It predicts a one-step-ahead value of utilization based on a fixed number of immediately preceding historical data measured at a constant-width time interval

The idea of the Mixed tendency-based prediction strategy is based on the assumption that if the current value increases, the next value will also increase, and if the current value decreases, the next value will also decrease.

The scheduling strategy arranges the workers based on the difference between the current and previous utilization values, Δu_t on an descending order. And then the longest chunk is assigned to the worker with smallest difference value.

5 Scheduling based on Average Prediction of Utilization technique APU

The main idea of our technique depends on calculating the average of the current and previous utilization values for each worker and assign the smallest chunk to the worker with maximum average value. The algorithm is shown below:

```
for j := 0 → m
   for i := 0 → n
      ave := ($u_{t-1}$ + $u_t$ ) / 2.0;
      utilization[i] := ave
      sort utilization in decreasing order
      sort the chunk array in increasing order
  for i := 0 → n
      assign chunk[i] to worker [i]
makespan ← the length of the longest queue
```

6 Experimental Results

A simulation is performed for the APU algorithm and the UMR algorithm to analyze the performance of each algorithm so that the APU performance can be compared with the UMR performance where the later is one of the most important algorithms for divisible load scheduling. The simulation is written using the C language. The model consists of n workers with n varying from 5 to 20. The round length is generated randomly using the gamma distribution function with beta varying from 500 to 3000. The speed of processors is chosen randomly. The values of bandwidth B_i, speeds and overheads $clat_i$ and $nlat_i$ for n=20 are shown in table 1.

Table 1 bandwidth B_i, speeds and overheads $clat_i$ and $nlat_i$

i	0	1	2	3	4	5	6	7	8	9
$Speed_i$	1	2	3	4	5	8	2	10	13	6
Bandwidth	2	3	4	5	6	5	3	6	11	12
$clat_i$	.3	.4	.5	.6	.7	.4	.5	.6	.8	.8
$nlat_i$	.4	.5	.6	.7	.8	.6	.9	.9	.5	.9

i	10	11	12	13	14	15	16	17	18	19
$Speed_i$	9	12	14	9	7	8	2	10	13	6
Bandwidth	6	13	9	19	16	5	3	6	11	12
$clat_i$	.9	.6	.6	.9	.9	.4	.5	.6	.8	.8
$nlat_i$	.7	.4	.8	.5	.5	.6	.9	.9	.5	.9

The local tasks is also generated using gamma distribution function with fixed arrival time lunda=.5 and beta = 50. To show the benefit of APU, Figure 3 plots the makespan versus different workload for both the Mixed tendency and APU Techniques.

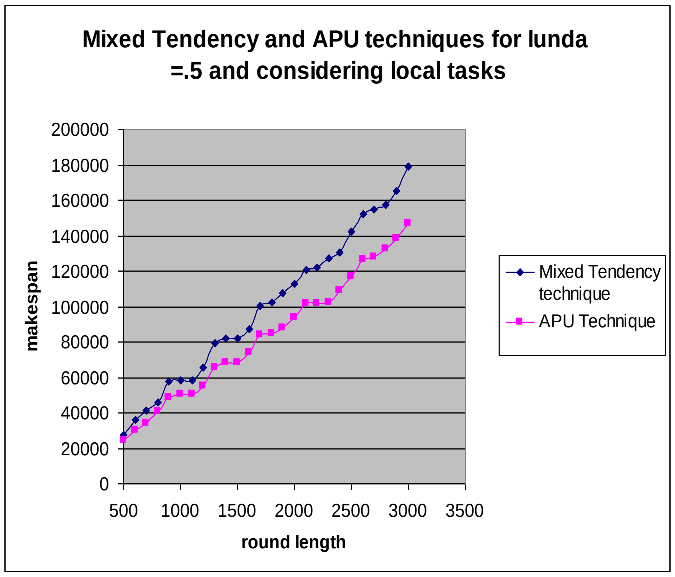


Figure 3 The makespan versus the round length

Figure 4 shows the effect of changing numbers of workers (n) on makespan produced by both the Mixed tendency and APU where beta=2500, lunda=.5 and n = 5, 10, 15, 20.

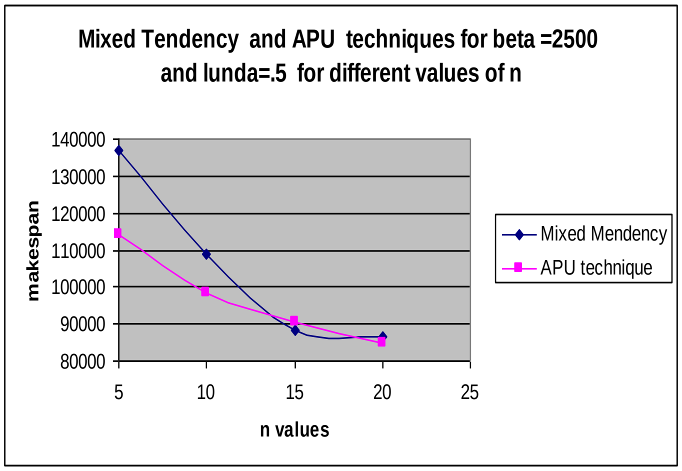


Figure 4 the makespan with different number of workers.

7 Conclusions

In this work, we proposed a scheduling strategy based on prediction method augmented to the UMR algorithm. The proposed method APU is compared with the linear way of prediction which compares values of current and previous values of utilizations to predict the next utilization. The results shows that the APU method reduces the makespan by 17.5% to 13% depends on the length of the chunk. Also, with different number of workers the makespan is improved by values from 16.5% to 2% with same workload. As a sketch of future work, we would try to apply the improved prediction method to other scheduling algorithms.

8 References

[1] Said ElNaffar and Nguyen The Loc, "Enabling Dynamic Scheduling in Computational Grids by Predicting CPU Utilization".

[2] Y. Yang and H. Casanova. "UMR: A Multi-Round Algorithm For Scheduling Divisible Workloads", Proceeding of the International Parallel and Distributed Processing Symposium (IPDPS'03), Nice, France, April 2003.

[3] Kim, HJ and Mani, V (2003), "Divisible load scheduling in single-level tree networks: Optimal sequencing and arrangement in the nonblocking mode of communication", Computers & Mathematics with Applications, 46 (10-11). pp. 1611-1623, 2009.

[4] O. Beaumont, H. Casanova, A. Legrand, Y. Robert, and Y. Yang, "Scheduling Divisible Loads on Star

and Tree Networks: Results and Open Problems", IEEE Transactions on Parallel and Distributed Systems (TPDS), 16(3):207–218, 2005.

[5] Y. Yang and H. Casanova, "Multi-Round Algorithm for Scheduling Divisible Workload Applications: Analysis and Experimental Evaluation", Technical Report CS2002-0721, Dept. of Computer Science and Engineering, University of California, San Diego, 2002.

[6] L. Yang, J.M. Schopf, and I. Foster, "Conservative Scheduling: Using Predicted Variance to Improve Scheduling Decision in Dynamic Environments", SuperComputing *2003*, Phoenix, Arizona USA November 2003.

[7] L.Yang, I. Foster, and J.M. Schopf, "Homeostatic and Tendency-Based CPU Load Predictions", *International Parallel and Distributed Processing Symposium (IPDPS'03)*, Nice, France, April 2003.