Selecting Best CPU frequency for energy saving in cluster using genetic algorithm

Conference Paper *in* IOP Conference Series Materials Science and Engineering · November 2020

DOI: 10.1088/1757-899X/928/3/032073

2 authors:

Ahmed Fanfakh
University of Babylon
16 PUBLICATIONS **49** CITATIONS
SEE PROFILE

Esraa H. Alwan
University of Babylon
5 PUBLICATIONS **5** CITATIONS
SEE PROFILE

PAPER • **OPEN ACCESS**

Selecting Best CPU frequency for energy saving in cluster using genetic algorithm

To cite this article: Zainab A. Abdulazeez *et al* 2020 *IOP Conf. Ser.: Mater. Sci. Eng.* **928** 032073

Selecting Best CPU frequency for energy saving in cluster using genetic algorithm

Zainab A. Abdulazeez, Ahmed B. M. Fanfakh and Esraa H. Alwan
Department of Computer science, College of science for women,
University of Babylon, Iraq

zainab.safar86@gmail.com, ahmed.fanfakh@uobabylon.edu.iq , esras.hadi@uobabylon.edu.iq

Abstract

Dynamic voltage and frequency scaling (DVFS) is a technique mainly used for reducing the consumed energy of computer's processor. Its only drawback detracting the performance of parallel application when executing them over parallel platform. However, genetic algorithm is introduced and applied in a heterogeneous cluster architecture for modeling the best trade-off between the energy saving and performance degradation of the parallel application in the same time. The suggested algorithm selects the best frequencies vector to achieve that targets by offering equal trade-off between them. The genetic algorithm simultaneously gives minimum energy consumption and minimal performance degradation via its objective function. All experiment will apply using SimGrid simulator. The experiments show that the algorithm reduces the energy consumption of the message passing application by (24% with 8000 problem size and 22% for 4000 problem size) and in almost cases improve the application performance up to 3%.

Keywords: Genetic algorithm, heterogeneous cluster, frequency scaling, energy consumption

1. Introduction

Computers consume large amount of energy while they run large application for solving bigger problems. Recently, acceleration is major concern and software were typically designed in such a way that the computations are performed as quickly as possible, resulting in a minimal execution time for the program. In particular, this approach is often used in parallel computing where a significant issue is a minimum execution time for the program. However, the intention of structuring program computations and memory accesses, several programming techniques and execution methods were developed in such a way to induce quick execution time. In the climate change era and, environmental concerns there is urgent need for efficient energy consumption in system performance. This is especially important for the implementation of massive, complex software systems in several areas of science and

of industry. Developing programming techniques for the system design is also essential, so that a hardware system can be leveraged computation performance with minimum energy consumption is guaranteed [1]. The most widely used tool for many processors of reducing the energy consumption of these platforms is the technique of dynamic voltage frequency scaling (DVFS). Its aimed to reduce dynamic power consumption by changing the frequency and voltage of CPU. This technique benefits from the fact that CPUs have discrete settings on voltage and frequency to decrease the processor frequency and thus saving its energy consumption. Therefore, this technique uses to decrease the total energy consumption of the heterogeneous architectures composing of various types of DVFS enabled processors [2]. This paper is structured accordingly as follows: section (2) presents related works to this paper and section (3) describes the predicting models of the execution time and the energy consumption of parallel MPI applications as a function of processor frequency scaling. Section (4) shows the predictions model of the energy consumption of parallel MPI applications based on processor frequency scaling. Sections (5) detail the proposed method to select the best frequencies in a heterogeneous cluster using genetic algorithm. Section (6) displays the results of experiments generated by the proposed method. Finally, section (7), presents the conclusions and future works.

2.Related work

In this section, some of the important works in the area of energy-performance optimizations are introduced.

Vaibhav [1] introduced a new energy reduction model for message passing iterative applications running on a heterogeneous platform. An online frequency selecting algorithm which has a small overhead and works without training or profiling is presented. This algorithm tries to give the best trade-off between energy saving and performance degradation for each node computing the message passing iterative application. The results show that in addition to limit the performance degradation, this algorithm can reduce the energy consumption by up to 34%. An analytical energy model that is simulating the energy consumption effect of frequency scaling was explored by [4]. In this model, the minimization of energy consumption to avoid waiting times at the joining points is achieved by using the scaling factor formula to set the frequency of the processors executing a task. For a particular system, a discrete set of the available levels of frequency scaling may use to adjust the optimal analytically scaling factors.

In [5] a weighted-sum approach for multi-objective optimization to schedule jobs on identical multiprocessors with shared memory architecture was addressed. GA-based method is used for optimizing energy consumption and performance for multiprocessor systems. Two different selection operators for performance optimization algorithm, which are the proportional roulette wheel selection (PRWS) and the rank-based roulette wheel selection (RRWS), is suggested. In addition, the effect of adding elitism in the GA was examined. In [7], One such study provides an energy-saving DEWTS scheduler depending on an algorithm for frequency scaling a dynamic voltage. DEWTS applies to the scheduling method of most data centers

composed of processors enabled by DVFS, which enables the tasks to be spread in idle slots at lower frequency and voltage without infringing the constraints of dependency and increasing the slacked make span.

An effective dynamic programming (DP) model based on Viterbi algorithm was proposed by [8]. It uses the Energy Delay Product (EDP) for an objective function and profiled information for applications to predict a best V/F rates to minimize execution times and energy consumption. The performance of the proposed algorithm was compared with greedy algorithm, an on-demand governor, and a feedback controller method. The reduction of the scheduling length while meeting the energy consumption constrained was introduce in [9]. This method transferring the energy consumption constraint of an application into tasks using a new task assignment method and a proof of which using mathematical induction. Instead of the ignorance of available energy in the MSLECC, the weighted average available energy of the tasks is used where the high performance is obtained without increasing time complexity. To validate the proposed WALECC algorithm, a large number of experiments with both fast Fourier transform and Gaussian elimination parallel applications was presented.

Finally, in [10] the performance-energy trade-off was shaped using learning-based algorithm. The lightweight contention metric that can efficiently convey the potential contention that will be faced by the workload was chosen in order to capture the contention among the co-runners. The model input is individual LLC aggressiveness, the global LLC intensity that emanates from the other cores and the performance constraint. The model output is the minimum core frequency at which the program will run to ensure the performance constraint that is provided by the co-runner effect.

3.Predicting the execution time of parallel MPI applications using processor frequency scaling.

This paper aims to reduce energy consumption of distributed iterative synchronous messaging applications running on a heterogeneous cluster. A heterogeneous cluster is described as a set of heterogeneous computing nodes linked through a homogeneous high-speed network. Hence, every node has different characteristics including frequency range of the CPU, computing power (FLOPS), energy consumption, while all its links connections have the same latency and bandwidth. The execution time for the synchronous distributed iterative application is calculated, which is the sum of the communication and the computation time of each iteration for each node. Moreover, consideration should be given to the possibility of a slack time occurring, which results in waiting for the fastest node of the slowest node until its computations are completed. Therefore, the total execution time is the time for the slowest task in the sense that it has the highest computation time without slack times [12].

DVFS (Dynamic voltage and frequency scaling) technology, which aims to reduce the energy consumption in CPU by lowering its frequency and voltage because DVFS scaling down the CPU frequency and subsequently its computing power. The execution time of the program running on the scaled down processor can be increased, the process of frequency reduction can be described by means of the scaling factor *S* that is the ratio between the largest frequency and the new frequency as set out in equation (1).

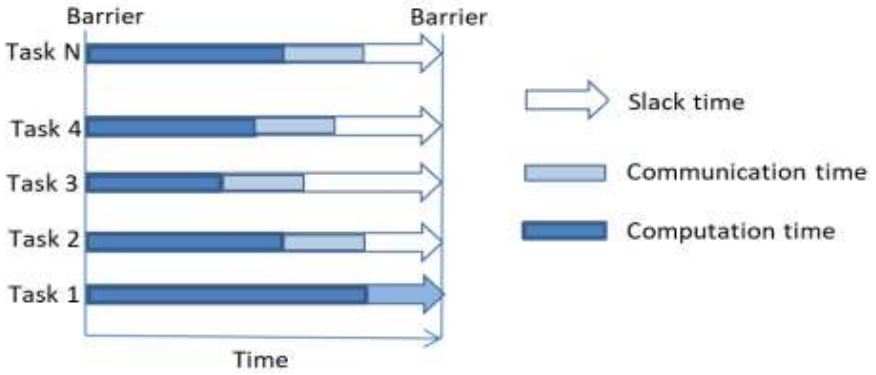

Figure.1 Parallel tasks execution over heterogeneous cluster.

$$S = \frac{F_{max}}{F_{new}} \quad \ldots. \ (1)$$

Since message passing distributed applications consists of two components: communication and computation, the scaling factor S has linear relationship with the execution time in relation to the computation time part, but the communication time is not affected by this parameter because the processors concerned remain idle while communication [3]. Since the platform is heterogeneous, then the nodes have different frequencies leading to different scaling factors (S0, S1, …, Sn) where *Si* is processor scaling factor of node *i*.

After the first iteration, the computation and the communication time of all tasks need to be calculated before any DVFS process can be used; in order to predict the time of execution of the message passing synchronous iterative applications running on the heterogeneous cluster for specific vector of scaling factor. The execution time for MPI of application for any scaling factor vector can be estimated by (2).

$$T_{new} = \underset{i=1,2,\ldots,n}{\text{Max}} \left(T_{cpOldi}.S_i\right) + MinT_{cm} \quad \ldots \ (2)$$

Where: $$MinT_{cm} = \min_{i=1,2,\ldots,n} T_{cm_i} \quad \ldots \ (3)$$

Where T_{cpOldi} is the processor's computation time i and $MinT_{cm}$ that is the slowest processor communication time. This model calculates the maximum time of

computation with scaling factor of slower node plus the communication time of slowest node.

4.Predicting the consumed energy of parallel MPI programs using processor frequency scaling.

In this section, the energy consumption by a processor is calculated, which is usually divided into dynamic and static energy. The dynamic energy is consumed during computation time, but the static one is consumed over all the times the processor is turn on [4]. Dynamic power was noted as *Pd* related to some parameters like the supply voltage V, load capacitance CL, the switching activity α and operational frequency F as shown in:

$$P_d = \alpha \,.\, CL \,.\, v^2 \,.\, F \ldots\ldots (4)$$

But the power static *Ps* that related with the leakage power computed as in equation (5).

$$P_s = V \,.\, N_{trans} \,.\, K_{design} \,.\, I_{leak} \ldots\ldots (5)$$

Where V is supply voltage, N_{trans} is a transistors number, K_{design} is a parameter dependent on design and I_{leak} is parameter dependent on technology. The energy that an individual processor consumes to execute the given program may be calculated as:

$$E_{ind} = P_d \,.\, T_{cp} + P_s \,.\, T \ldots\ldots (6)$$

Where T is the program execution time and T_{cp} is computation time, thus T_{cp} < T. While there is no communication and no slack time T_{cp} may be equal to T.

DVFS operation's main purpose is to minimize total energy consumption [13]. The operating frequency *F* is linearly dependent on the supply voltage V, i.e. V=β.F with constant β, hence the new F_{new} frequency from the equation (1) is calculated as follows:

$$F_{new} = S^{-1} \,.\, F_{max} \ldots\ldots (7)$$

By replacing F_{new} in (4) like in (7), for dynamic power consumption the following equation is given:

$$p_{dNew} = \alpha \,.\, C_L \,.\, V^2 \,.\, F_{new} = \alpha \,.\, C_L \,.\, B^2 \, {F_{new}}^3$$

$$= \alpha \,.\, C_L \,.\, V^2 \,.\, F_{max} \,.\, S^{-3} = P_{dOld} \,.\, S^{-3} \ldots\ldots (8)$$

Where P_{dOld} and P_{dNew} are the dynamic power consumed as a function to the maximum frequency and the new frequency respectively. Therefore, a factor of S^{-3} reduces the dynamic power when the frequency is lowered by factor *S* [14]. Since a

CPU's FLOPS is proportional to a CPU's frequency, the computation time is increase in proportion to factor *S*. A new dynamic energy is the dynamic power which is multiplied by the new computational time as follows:

$$E_{dNew} = P_{dOld} . S^{-3} . \left(T_{cp} . S \right) = S^{-2} . P_{dOld} . T_{cp} \dots \dots (9)$$

Static energy was the product of the execution time of the application by the static power as described in the model of execution time in the equation (2). Program's execution time is sum of the communication and the computation times that linearly related with the frequency scaling factor with no impact on the communication time. Processor's static energy is determined as follows after scaling its frequency:

$$E_s = P_s . \left(T_{cp} . S + T_{cm} \right) \dots \dots (10)$$

Therefore, in the case of the heterogeneous platform, each processor *i* has specific static and dynamic power noted respectively as P_{si} and P_{di}. A processor's communication time is indexed as T_{cm_i} and may contain slack times when slower nodes are communicated. There are no equal waiting times for all nodes but each CPU *i* has different computation time indexed as T_{cp_i} and various frequency scaling factors can be measured to reduce the total energy consumption and the slack time of application. While the static energy was calculated as a sum of the execution time multiplied by each processor's static power, the dynamic energy is measured by the frequency scaling factor and also the dynamic power as given in (9) for each node. The energy consumption for each processor is sum of the static and dynamic energies which is determined as follow:

$$E = \sum_{i=1}^{N} \left(S_i^{-2} . P_{d_i} . T_{cp_i} \right) + \sum_{i=1}^{N} \left(P_{s_i} . \left(\max_{i=1,2,\dots,N} \left(T_{cp_i} . S_i \right) + MinT_{cm} \right) \right) \dots (11)$$

Reducing processor frequencies according to the scaling factor vector (S1,S2, ..., SN) can degrade application performance and therefore increasing the static energy due to an increase in the execution time [12].

5. Selecting the best frequencies in a heterogeneous cluster using genetic algorithm

Indeed, when scaling down a processor's frequency the dynamic power is decreased, but its computing power is reduced proportionally. Therefore, the execution time may be significantly increase. Whereas, both static and dynamic powers are consumed during that time are increased too. Thus, the overall application energy consumption may not be the optimum one. In heterogeneous parallel architecture, both dynamic and static powers are different and the ratio of computation to communication times are different. Consequently, the process of choosing the best frequency for each processor not easy. Therefore, the scaling factors vector according to the best trade-off between the consumed energy and performance

should be chosen. The goal of this work is to reduce the consumed energy and avoiding the increase in the running time significantly. The relationship between the execution time and the energy consumption for an application is hard and nonlinear. To resolve this problem, firstly it normalizes the running time through calculating the ratio between a new and the old execution time, as follows:

$$P_{Norm} = \frac{T_{New}}{T_{Old}}$$

$$= \frac{max_{i=1,2,\dots,N}\left(T_{cp_i} . S_i\right) + MinT_{cm}}{max_{i=1,2,\dots,N}(T_{cp_i} + T_{cm_i})} \dots . \quad (12)$$

Similarly, the energy is normalized by measuring the ratio between both the consumed energy when the frequency is scaled down and the energy consumed at maximum frequency for all nodes as follows:

$$E_{Norm} = \frac{E_{Reduced}}{E_{Orginal}}$$

$$= \frac{\sum_{i=1}^{N} \left(S_i^{-2} . P_{d_i} . T_{cp_i} \right) + \sum_{i=1}^{N} \left(P_{s_i} . T_{New} \right)}{\sum_{i=1}^{N} \left(P_{d_i} . T_{cp_i} \right) + \sum_{i=1}^{N} \left(P_{s_i} . T_{Old} \right)} \dots (13)$$

where the main goal is to achieve maximum reduction of energy with minimal reduction of the execution time, but according to equations (12) and (13), the frequency scaling factor vector simultaneously decreases both energy and execution time. To solve this problem, making the execution time and energy optimization process follow a same evolution according the scaling factors vector. However, inverting the normalized execution time equation to give the normalized performance equation as follows:

$$P_{Norm} = \frac{T_{Old}}{T_{New}}$$

$$= \frac{max_{i=1,2,\dots,N} (T_{cp_i} + T_{cm_i})}{max_{i=1,2,\dots,N} \left(T_{cp_i} . S_i \right) + MinT_{cm}} \dots . (14)$$

The objective function could then be modeled on all available scaling factors in order to calculate at the same time the maximum difference between energy normalization and performance normalization. In other words, it represents the minimum energy consumption and maximum performance at the same time, and the objective function computed by the following equation:

$$MaxDist = \max_{\substack{i=1,\dots,F \\ j=1,\dots N}} (P_{Norm} \left(S_{ij}\right) - E_{Norm} (S_{ij})) \dots (15)$$

Where the number of nodes is *N*, and number of frequencies available for each node is *F*. Then it can select the optimum set of scaling factors that satisfying equation (15).

To solve the problem of idle or slack time due to time differences between fastest and slowest node due to the heterogeneity in the sense that each node has a different

computation time than the other, the initial frequency scaling factors computes as follows:

$$S_{cp_i} = \frac{max_{i=1,2,\dots N}\left(T_{cp_i}\right)}{T_{cp_i}} \dots (16)$$

Depending on the initial frequency scaling factors, all nodes compute the initial frequencies as follows:

$$F_i = \frac{F max_i}{S_{cp_i}},\ i = 1,2,..,\text{N} \dots (17)$$

If a frequency calculated of the nodes is not available in its node frequency vector, the closest frequency available to it is chosen. Thus, the frequencies of the fastest nodes will be reduced depending on the scaling factors for the frequencies and this reduces the search space for selecting best frequency vector, since higher frequencies increase energy consumption and do not improve application performance. Therefore, the process of searching for frequency scaling factors will start from these initial frequencies downward towards the lower frequencies and thus we get optimum frequency scaling factors which give the highest value to the object function.

In this work, the genetic algorithm is applied to solve the problem of selecting the best CPU frequencies. Genetic algorithms are a search tool used to optimize the candidate solution based on the principles of evolution [5]. The algorithm starts its search from a random generated vectors CPU's frequency, where each vector is the candidate solution in the evolutionary process. Genetic operator including selection, crossover and mutation are applied over the generated population solutions to compose the next population. The next genetic algorithm determines the frequency scaling factors vector to provide the good possible trade-off between reducing energy consumption. It maximizes the performance, depending on the equation (15), of an iterative program uses message passing interface executing on the heterogeneous cluster. It uses the information collected offline after running the parallel program like the communication and computation time.

Algorithm

Require

T_{cp_i} : *array for all computation times during first iterations and for all nodes with the highest frequency.*

T_{cm_i} : *array for all communication time during the first iteration and for all nodes with highest frequency.*

Popsize: size of population

Clustersize: maximum number of nodes in cluster.

Maxgen: Maximum Generation Number

P_{d_i}: *array for dynamic array of all nodes.*

P_{s_i}: *array for static array of all nodes.*

F_{max_i}: *array for maximum frequency of all nodes.*

F_{min_i}: *array for minimum frequency of all nodes.*

1: $S_{cp_i} = \frac{max_{i=1,2,\ldots,N}(T_{cp_i})}{T_{cp_i}}$

2: $F_i = \frac{F_{max_i}}{S_{cp_i}}$

3: *Round every initial computed frequency Fi for the nearest one in each node available.*

4: *Generate the initial population by selecting random frequencies for each solution in the range Fi to*

F_{min_i}

5: $T_{Old} = max_{i=1,2,\ldots,N}\ (T_{cp_i} + T_{cm_i})$

6: $E_{original} = \sum_{i=1}^{N}(P_{d_i} . T_{cp_i}) + \sum_{i=1}^{N}(P_{s_i} . T_{Old})$

7: Gen ← 0

8: Repeat

9: Gen ← Gen +1

10: for i=1 to popsize do

11: for j= 1 to clustersize do

12: $S_i = \frac{F_{max_i}}{F_i}, i = 1,2,\ldots,N.$

13: $T_{New} = max_{i=1,2,\ldots,N}\left(T_{cp_i} . S_i\right) + MinT_{cm}$

14: $E_{Reduced} = \sum_{i=1}^{N}(S_i^{-2} . P_{d_i} . T_{cp_i} + P_{s_i} . T_{New})$

15: $P_{norm} = \frac{T_{orginal}}{T_{predicted}}$

16: $E_{norm} = \frac{E_{predicted}}{E_{orginal}}$

17: $Fitness_i = P_{Norm} - E_{Norm}$

18: end for

19: end for

20: Selecting the individuals by using tournament selection method

21: Applying crossover over the selected parents

22: Applying mutation over the two new parents.

23: Evaluation of the new offspring using step from 10 to 17

24: Replace the best individual of the new generation with the worst individual of the randomly chosen group.

25: until (Gen >=Maxgen) or (the best individual not change for a given number of generations)

6. Experiments

6.1 Experiment setting

To validate the proposed method, this work uses Linux operating system and message passing library MPI to program and execute the proposed method. The proposed method has been applied to the Jacobi method by using the SimGrid / SMPI simulator that provides easy and strong tools for simulating heterogeneous cluster. It consists of four types of nodes with different characteristics. The heterogeneous

cluster included nodes of the four types, where similar number of nodes uses for each type. The algorithm operates offline, after completion of running the parallel iterative program. The genetic algorithm gathers the required information for the application when its finish running. The gathered information for each node in the cluster is computation time, communication time, frequencies, static and dynamic power. Table(1) shows the characteristics of the four nodes types in the cluster. The algorithm predicts the energy consumption and time of execution for any vectors of frequency scaling factor, based on all of that information. The algorithm was repeated several times until the best vector for the frequency scaling factors was determined, which results will be given in the next section.

Table 1: nodes information

Node type	Max freq.	Dynamic power	Static power
1	2.5	20 w	4 w
2	2.6	25 w	5 w
3	2.8	30 w	6 w
4	3.4	35 w	7 w

Some of the important features that genetic algorithm distinguishes the use of crossover and mutation from other evolutionary algorithms, these processes are performed according to a certain probability called crossover probability and mutation probability, defined as PC and PM consecutively. The probabilities range is defined such as $0 < PC < 1$, $0 < PM < 1$ [11]. Thus, the proposed algorithm applied on different number of nodes starting from 4,8,16,32 till 64 nodes. Moreover, nine values of PC and PM as shows in table (2) and three population sizes (100, 200 and 300) are used in the experiments.

Table (2): value of probabilities

PC	1	1	1	0.9	0.9	0.9	0.8	0.8	0.8
PM	0.1	0.2	0.3	0.1	0.2	0.3	0.1	0.2	0.3

6.2 Experiment results

The model of energy consumption depends mainly on the model of time of execution since the static energy related linearly to the time of execution but the dynamic energy was relative only to time of computation. However, to verify of the efficiency of the proposed algorithm, the actual execution time was compared with a predicted execution one, as well as the real and predicted consumed energy are compared. The comparison indicated that the model proposed is precise, and as maximum difference between the time of execution and predicted time was less than

10%. For each instance, the total energy consumption was determined according to the model of energy consumption (11), with and without the algorithm being implemented. In all these experiments, the execution time was also calculated. Then, the percentages of energy savings and performance degradation were calculated for each instance. The experiments demonstrate that the proposed algorithm noticeably minimize energy consumption (to 24.78% if the problem size is 8000 and 22.66% for problem size is 4000) and attempts to limit the degradation of performance to minimum. Results are shown in tables (4) , (5), (6), (7), (8) and (9) were the best scenarios for the three populations size with different probabilities for crossover and mutation. All these results are the average energy-saving and performance-degradation values from several experiments. It's often show that the percentage of energy savings decreases as the number of estimated nodes increases. That decrease is due to increase in the communication times when the application is running on the higher number of nodes. By comparing the execution times, but results show that percentages of performance degradation of most instance decrease when operating on the large number of nodes. This is because using more-nodes will expend less time of computing than communication time. Therefore, reducing the frequencies of some of these nodes reducing the performance too.

The obtained results show that the performance degradation ratio has positive and negative values. The positive values refer to performance degradation while the negative ones refer to the improvement in the performance compare to execution time of cluster without DVFS. Table (10) presents the details of the best scenarios selected from all obtained results. Figures 2 and 3 show the results of the best scenarios for two problem sizes which are 4000 and 8000.

Table (4): best scenario- population size=300 problem size=4000

Nodes	Scenario	Energy-saving	Performance Degradation
4	pc=0.9,pm=0.2	21.3413	-1.36513
8	pc=0.9,pm=0.3	21.32557	-3.71017
16	pc=0.9,pm=0.3	21.98475	-4.05505
32	pc=0.8,pm=0.1	20.89991	-2.07293
64	pc=0.8,pm=0.1	15.80963	-0.5272

Table (5): best scenario- population size=200 problem size=4000

Nodes	Scenario	Energy-saving	Performance Degradation
4	pc=1,pm=0.3	21.8226	-5.0497
8	pc=0.9,pm=0.3	21.1444	-3.7095
16	pc=0.8,pm=0.2	22.6639	-0.7734
32	pc=0.9,pm=0.3	18.8237	-0.4349
64	pc=1,pm=0.2	18.00774	-3.7327

Table (6): best scenario- population size=100 problem size=4000

Nodes	Scenario	Energy-saving	Performance Degradation
4	pc=0.9,pm=0.1	21.7442	-1.9559

8	pc=1,pm=0.2	20.5447	-0.4822
16	pc=1,pm=0.1	21.5739	-3.1389
32	pc=0.8,pm=0.3	20.1068	-2.0765
64	pc=1,pm=0.2	16.4663	-2.3473

Table (7): best scenario- population size=100 problem size=8000

Nodes	Scenario	Energy-saving	Performance Degradation
4	pc=0.9,pm=0.2	22.89169	-3.72779
8	pc=0.9,pm=0.3	20.57708	-2.6553
16	pc=0.9,pm=0.3	19.67462	-0.13292
32	pc=0.8,pm=0.3	22.42769	-4.89822
64	pc=1,pm=0.1	19.45389	-5.99994

Table (8): best scenario- population size=200 problem size=8000

Nodes	Scenario	Energy-saving	Performance Degradation
4	pc=0.9,pm=0.3	21.52145	-1.80212
8	pc=0.9,pm=0.1	21.89779	-2.97942
16	pc=0.8,pm=0.1	21.82149	0.851252
32	pc=0.8,pm=0.1	24.86672	-6.14255
64	pc=1,pm=0.2	21.25644	-7.00692

Table (9): best scenario- population size=300 problem size=8000

Nodes	Scenario	Energy-saving	Performance Degradation
4	pc=0.8,pm=0.2	24.77813	0.365168
8	pc=0.8,pm=0.3	20.59369	-3.13361
16	pc=1,pm=0.1	21.65839	-2.25353
32	pc=0.8,pm=0.1	24.95171	-5.04255
64	pc=0.8,pm=0.2	19.8148	-5.26921

Table (10): Best scenarios information for specific number of nodes

Problem size	Scenario details			Number of nodes	Its name
	Pop-Size	PC	PM		
4000	100	0.9	0.1	4	S1-4000
	200	0.8	0.2	16	S2-4000
	300	0.9	0.3	16	S3-4000
8000	100	0.9	0.2	4	S1-8000
	200	0.8	0.1	32	S2-8000
	300	0.8	0.1	32	S3-8000

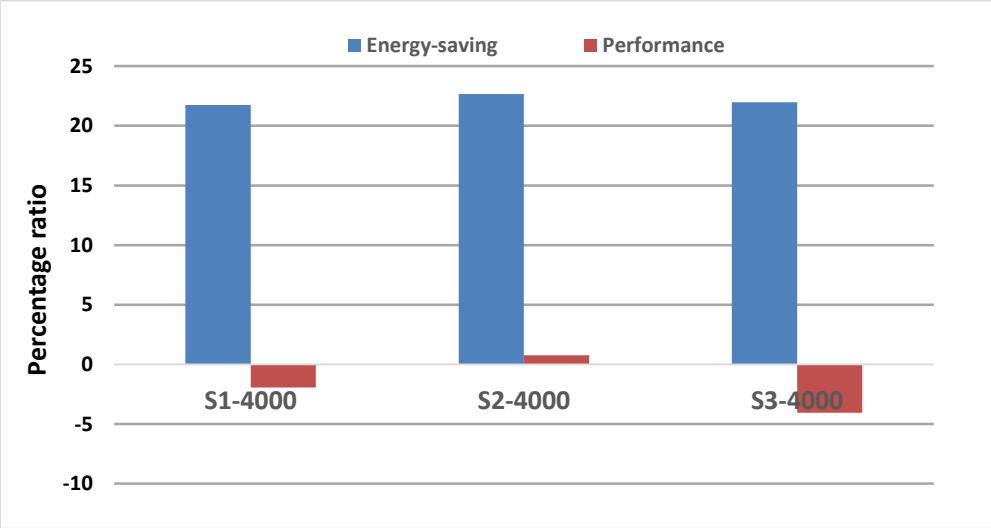

Figure (2) : The best scenario result of population size 4000

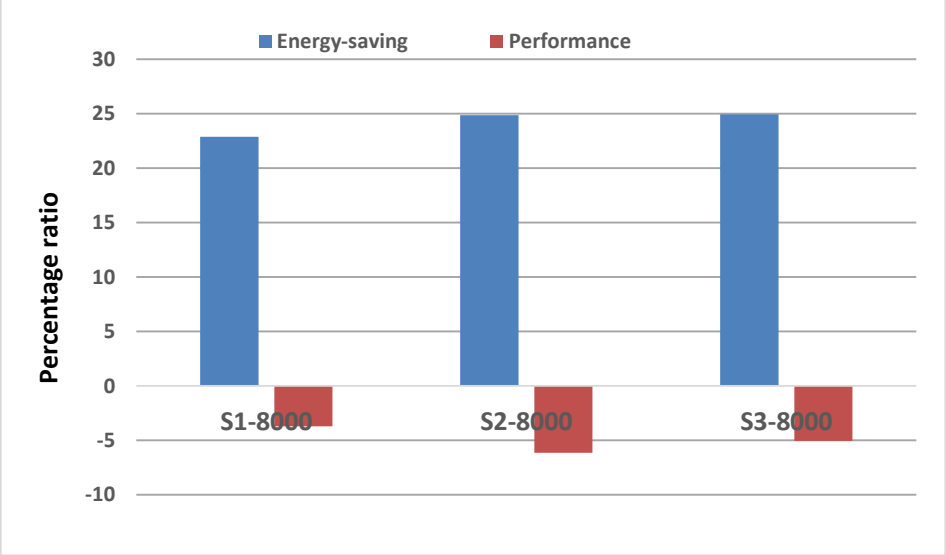

Figure (3) : The best scenario result of population size 8000

7 .Conclusions

This work introduced an algorithm for selecting the best frequency vector in the distributed cluster platforms. It provides the minimum energy consumption for iterative distributed applications running on heterogeneous cluster with minimal degradation in performance. This algorithm selects various scaling factor vectors based on the ratio of communication and computing times and the values of the CPU's dynamic and static power. The result of an experiments presented that the algorithm minimizes the consumed energy of

the parallel iterative application to (24% with 8000 problem size and 22% for 4000 problem size) while in almost cases improve the application performance up to 3%.

References

[1] Vaibhav Sundriyal, Masha Sosonkina, Modeling of the CPU Frequency to Minimize Energy Consumption inParallel Applications, Sustainable Computing: Informatics and Systems,2017.
[2] Nikzad Babaii Rizvandi , Javid Taheri , Albert Y. Zomaya, Some observations on optimal frequency selection in DVFS-based energy consumption minimization. Journal of Parallel and Distributed Computing Volume 71, Issue 8, August 2011, Pages 1154-1164.
[3] Jean-Claude Charr , Raphael Couturier, Ahmed Fanfakh and Arnaud Giersch, Consumption Reduction with DVFS for Message Passing Iterative Applications on Heterogeneous Architectures. The 16 th IEEE International Workshop on Parallel and Distributed Scientific and Engineering Computing. pp. 922-931. IEEE Computer Society, INDIA (2015).
[4] Thomas Rauber and Gudula Runger. Analytical modeling and simulation of the energy consumption of independent tasks. In Proceedings of the Winter Simulation Conference, pages 245:1-245:13. Winter Simulation Conference, 2012.
[5] Anju S. Pillai, Kaumudi Singh, Vijayalakshmi Saravanan, Alagan Anpalagan, aIsaac Woungang and Leonard Barolli. A genetic algorithm-based method for optimizing the energy consumption and performance of multiprocessor systems, Springer-Verlag GmbH Germany 2017.
[6] N. Kumar and D. P. Vidyarthi , A GA based energy aware scheduler for DVFS enabled multicore systems ,Department of Computer Science and Engineering, ABV-Indian Institute of Information Technology & Management, Gwalior 474015, India,2017.
[7] Zhuo Tang, Ling Qi, Zhenzhen Cheng, Kenli Li, Samee U. Khan and Keqin Li, An Energy-Efficient Task Scheduling Algorithm in DVFS-enabled Cloud Environment, Springer Science+Business Media Dordrecht 2015.
[8] Shervin Hajiamini, Behrooz Shirazi, Aaron Crandall and Hassan Ghasemzadeh, A Dynamic Programming Framework for DVFSbased Energy-Efficiency in Multicore Systems, IEEE Transactions on Sustainable Computing 2019.
[9] Fengsong Hu, Xiajie Quan and Can Lu, A Schedule Method for Parallel Applications on Heterogeneous Distributed Systems with Energy Consumption Constraint, 2018, Shenzhen, China.
[10] Solomon Abera, M. Balakrishnan, and Anshul Kumar, Performance-Energy Trade-off in CMPs with Per-Core DVFS, Springer International Publishing AG, part of Springer Nature 2018.
[11] Ali, M. Z., Awad, N. H., Suganthan, P. N., Shatnawi, A. M., & Reynolds, R. G. (2018). An improved class of real-coded Genetic Algorithms for numerical optimization.Neurocomputing, 275, 155–166. doi:10.1016/j.neucom.2017
[12] Idrees S.K., Fanfakh A.B.M. (2018) Performance and Energy Consumption Prediction of Randomly Selected Nodes in Heterogeneous Cluster. In: Al-mamory S., Alwan J., Hussein A. (eds) New Trends in Information and Communications Technology Applications. NTICT 2018. Communications in Computer and Information Science, vol 938. Springer, Cham.
[13] Fanfakh, A., Charr, J., Couturier, R. et al. Energy consumption reduction for asynchronous message-passing applications. J Supercomput 73, 2369–2401 (2017).
[14] A. Fanfakh, J. Charr, R. Couturier and A. Giersch, "CPUs Energy Consumption Reduction for Asynchronous Parallel Methods Running over Grids," 2016 IEEE Intl Conference on Computational Science and Engineering (CSE) and IEEE Intl Conference on Embedded and Ubiquitous Computing (EUC) and 15th Intl Symposium on Distributed Computing and Applications for Business Engineering (DCABES), Paris, 2016, pp. 205-212, doi: 10.1109/CSE-EUC-DCABES.2016.186.