Study the impact of using Lock-free buffer to communicate the DOACROSS loop iterations

Esraa H.A. Alwan
Computer science
College of science for women -Babylon University
wsci.israa.hadi@uobabylon.edu.iq
Isr.phd@gmail.com

Abstract

Communication dependency overhead becomes the biggest obstacle that facing the parallelizing loops containing loop-carried dependencies such us DOACROSS loop. Although of a substantial researches had been devoted to this field, the problem still far from solved.

This work introduces a FastForward circular lock-free queue algorithm to communicate the dependency between DOACROSS loop iterations. Instead of giving each iteration of DOACROSS loop to thread as in the original methods, group of iterations will be given to each thread. So to ensure correct results, the dependence between threads must be respected and for parallelism to be effective, the overhead on core-to-core communication must be as low as possible.

Experimental results are implemented on Intel Core i7 processor that has 4GB RAM running SUSE operating system show performance improvements of the proposed DOACROSS approach. An evaluation of this technique on four programs with a range of dependence patterns lead to ≈ 0.9 speed up.

Keywords: DOACROSS, Lock-free buffer, Parallelizing techniques, Multicore

الخلاصة

الكلفة العالية لنقل المعلومات بين المعالجات في الحاسبات المتعددة المعالجات هي واحدة من اكبر المشاكل التي تواجه تنفيذ الدورات ذات الاعتمادية بصورة متوازية ، على الرغم من الجهود الكبيرة التي بذلت من في هذا المجال الا ان المشكلة مازالت بعيده عن الحل. هذا البحث يدرس تأثير استخدام خوارزمية الطابور الدائري بدون القفل لنقل المعلومات بيت المعالجات. لضمان صحة النتائج فان الاعتمادية بين التكرارات الخاصه بالزوارق يجب ان تراعى بالاضافة الى ان كلفة نقل المعلومات يجب ان تكون قليله قدر الامكان. النتائج التي تم الحصول عليها من التجارب تم تنفيذها على حاسبة من نوع Core i7 . تنفذ نظام تشغيل سوري ان التقنيه المقترحة قامت بتحسين الأداء وأدت الى تحسين وقت التنفيذ بنسبه 0.9

الكلمات المفتاحية : الخزان المؤقت بدون مفتاح, تقنيات التنفيذ المتوازي, الحاسبات المتعددة المعالجات

I. Introduction

Raising the sequential application performance required many improvements in both commodity hardware systems and compiler optimization techniques. A range of microarchitectural techniques that have been used to enhance the performance of single thread applications in a highly effective way. These techniques include the superscalar issue, out-of-order execution, on-chip caching, and deep pipelines supported by sophisticated branch predictors (Spracklen and Abraham,2005; Rangan,2008).

A new strategy has been introduced by processor designers which involved steadily increasing in the number of transistors, through which many cores are placed in one chip to replicate the performance of a single faster processor.

Multicore systems have become a dominant feature in computer architecture. Chips with 4, 8, and 16 cores are available now and higher core counts are promised. Unfortunately single-threaded legacy application does not achieve better performance

when it is executed on multicore system. However the level of gain becomes higher with software techniques which are parallelizing the sequential application to get better performance (Matthew Bridges,2008; Adve,2010).

Parallelizing single threaded application by converting it to multi-threaded application that can be used most of the cores in multi-core architecture become the most important current research topics. Traditional parallelizing techniques such as DOALL and DOACROSS work through distribution the loop iterations among many cores (Matthew Bridges,2008).While the DOALL techniques improve the program performance when applied in the numerical field, the overhead of communicating the dependency between threads in DOACROSS technique is so large and can be negated any advantage due to the lack of hardware support for inter-core communication.

In this paper, we study the impact of using a Fastforward lock-free buffer method on DOACROSS technique. Fastforward circular lock free buffer has been used to communicate the dependency between thread without any lock where the producer and consumer can reach the queue concurrently. Unlike the a lock-based approach, such as pthread mutex lock which can negate any benefit from the parallelizing due to their overhead. The rest of this paper is structured as follows: the next section (II) introduces the background for DOACROSS technique, then section III explain how can be using the Fastforword lock free buffer to parallelize the DOACROSS technique. Section IV shows the implementation of the proposed method. V presents some experimental results from the application of the manual transformation. Finally in section VI, we survey related work and conclude .

II. Background

DOACROSS is the most popular Cyclic Multi-Threading (CMT) transformation. This technique tries to execute the loop body in parallel even in the exist the cross dependency between it is iterations. This technique works similarly to DOALL, see figure 1, by giving each iteration to a thread and these threads are executed on multi-core in round-robin fashion. In contrast with the DOALL technique, however, there are data and control dependencies crossing loop iteration boundaries.

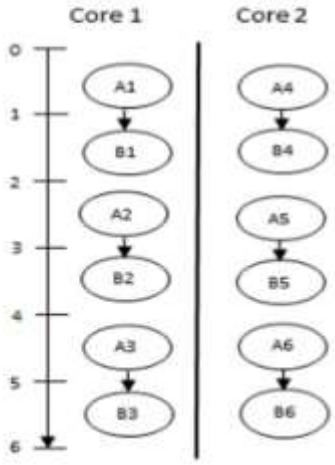

Figure 1: DOALL technique. Adapted from (Raman,2009)

Therefore, to get correct results the dependency should be respected and the synchronization should be in the right order so as to ensure that the following iteration receives the correct value (Chen,2010; Rajamony,1997). With the increased number of cores integrated on the same die, the communication latencies between them become more significant, with their values ranging from a few tens of cycles to even a few hundreds. As a result, this leads to reduced performance of CMT, for unlike with Independent Multi-Thread (IMT) the increased number of threads in the system does not always lead to a linear increase in performance. Figure 2 shows an example of this technique. DOACROSS schedules each loop iteration on an alternate thread and communicates the dependency from thread to thread in a cyclic fashion due to the pointer chasing in statement A. Set of odd iteration will be given to the first thread while the rest will be given to the second one (Unnikrishnan et al.,2012;Zhong,2001). The total time to execute the DOACROSS parallelized loop equals:

$$L_{par} = \frac{n}{m} * (L_i + SC) \qquad (1)$$

where n represents the number of loop iterations, m the number of available threads, *iter*1, *iter*2, *iter*3, *iter*4 represent loop iterations, Li the execution time for one loop iteration