Advances in Computer Science Research, volume 50
4th International Conference on Electrical & Electronics Engineering and Computer Science (ICEEECS 2016)

The algorithm of thread scheduling with an Improvement Accelerating Critical Section of migration strategy

Xueming Zhai[1, a], Xu Li[2, b]

[1]Institute of control and computer engineering, NORTH CHINA ELECTRIC POWER UNIVERSITY,baoding,071000,China

[2]Institute of control and computer engineering, NORTH CHINA ELECTRIC POWER UNIVERSITY,baoding,071000,China

[a]email: xu1905963046@163.com, [b]email:361406379@qq.com

Keywords: Critical Section Migration; Priority Factor; Dynamic Scheduling

Abstract. In multi-core processors,the heterogeneous multi-core processors are more practical than homogeneous multi-core because they have asymmetric cores.Therefore,the multi-thread scheduling among them becomes a hot research topic.In this paper, an Improvement Accelerating Critical Section algorithm is proposed for the improvement of the Accelerating Critical Section algorithm. The algorithm tracks the execution of threads on the cores that contain the critical section and changes the priority factor of threads .It use the priority factor to guide the threads'dynamic migration between the high-performance core and the low-performance cores.The algorithm can accelerate the excution of the critical section and effectively solve the threads block problem on high-performance core that the Accelerating Critical Section causes.This paper use the Simics to simulate heterogeneous multi-core processors system and the inputs are 12 loads with intensive critical sections. The experimental results show that compared to the Accelerating Critical Section algorithm and the Simple Algorithm ,the algorithm this paper proposes has better performance.

Introduction

Due to restrictions such as power consumption, development level of the semiconductor technology,now the processor is enhanced by multi-core technology to simply rely on raising the main frequency .Multi-core refers to the integration of multiple processor cores on a single chip.And on-chip multi-core processor (CMP) is divided into homogeneous multi-core processora and heterogeneous multi-core processors.

Homogeneous multicore processors refer to the processors with all same cores but it is different to heterogeneous multi-core processors on performance and power consumption between different cores.Homogeneous multi-core processor has same cores.However,the demand for core resources among different threads and the various stages of the demand for resources are constantly changing [1] [2], so the homogeneous multi-core processor will cause the waste of core resources. Heterogeneous multi-core processor has different cores,so it can adapt to the various threads and thread's each stage of the differences and it is more effective in using resources. Compared to a homogeneous multi-core processor ,a mixed cores heterogeneous multi-core processor has more advantages and can better meet the different performance and functional requirements, and is the trend of the future development of multi-core processor. Nowadays, more and more scholars have focused on the heterogeneous multi-core thread scheduling [3] [4]. Good scheduling algorithm can better utilize the resources of multi-core processors, and the thread scheduling on the heterogeneous multi-core processors is even more complicated.

The scheduling algorithm for multi-core processor [5] is generally divided into two categories, one is static scheduling method the other on is dynamic scheduling method. Static task scheduling is performed by the compiler at compile time, such as based on the list structure algorithms, clustering algorithm and based on the copy algorithms. Dynamic task scheduling is based on the actual running situation of tasks in the system to schedule to the corresponding computing unit to execute, and at the same time meet the requirements of the system constraints. Because the static scheduling method cannot dynamically take the change of the thread and the change of the processor into account , the dynamic method is more practical than the static method.

Because threads cannot update shared memory at the same time, the operation of shared data is encapsulated in the critical section.A critical section is a program fragment that accesses shared resources, such as shared devices or shared memories, that cannot be accessed by multiple threads at the same time.When a thread enters the critical section, the other threads or processes must wait, some synchronization mechanism must be used in the the entry and exit points of program to ensure that these shared resources is mutually exclusive used.At a given time, only one thread can execute the critical section.Other threads that want to execute the critical section can only wait.The critical section can cause the thread serialization, thus affecting the performance.Reducing the execution time in the critical section can reduce the decrease of this performance.An accelerating idea is to migrate the critical area to the high speed core, and using the high performance core to shorten the time of the critical section.But this method brings about a series of crtical sections [6] that are not relevant to each other.

In this paper, the IACS algorithm is proposed to improve the Accelerate Critical Section (ACS) algorithm.The algorithm tracks all the critical sections of the error of the serial behavior, the algorithm defines the priority factor, and gives the high priority factor to the threads that frequently rejected on the high speed core.

Improved critical section migration strategy

A. Introduction of ACS Method

Because the thread cannot update the shared data simultaneously，instruction to access shared data is encapsulated in a critical area.At a given time，only one thread can execute the critical section，other threads that need to perform critical section can only wait.The critical section can cause the thread to be serial,which leads to a reduction in performance.Reducing the execution of the critical section will reduce this loss.

In the traditional system,when kernel is in the execution of a critical section of the thread，it gets the lock of the critical section.In ACS，when a low speed core encounter critical section，it sends a request to the high speed core and suspended execution.When the high speed core is completed and the low speed core continues to be executed.ACS reduces serialization by accelerating the execution of the critical region,shorten the time of thread waiting for critical section.However,The execution of the critical section on the high speed core will cause some problems.ACS can result in a series of critical secitons' serialization which are independent of each other and can be executed in parallel.We call this phenomenon the error of serialization.We can solve this problem by using SMT (simultaneous multi -threading) technology adding virtual high speed core.However, sometimes it can not completely avoid serialization error.In this paper, we propose the improved algorithm IACS for ACS.

B. IACS System Structure

IACS technology is applied in a single instruction set，heterogeneous CMP，and provides hardware support for cache consistency.ACS based on asymmetric multi-core processor (ACMP).ACMP contains at least one high-speed core and a number of low speed cores.ACMP was originally proposed to perform Amdahl serial bottleneck， uses high speed core to execute serial sections and low speed core to execute parallel sections.

Figure 1 shows an application of IACS on the ACMP structure ,which contains 1 high speed cores (P0) and 12 low speed cores(P1-P12).Same as ACS, there is a critical area request buffer (CSRB) in P0 to receive requests from low speed cores.And gives CSRB a priority factor table to record the priority factor in the critical section.IACS’s two instructions CSCALL and CSRET are respectively inserting to the beginning and end of the critical section program.CSCALL contains two messages:LOCK_ADDR and FIN_ADDR are used to record the address of the lock in the critical section and the address of the first instruction in the critical section.CSRET storage LOCK_ADDR.

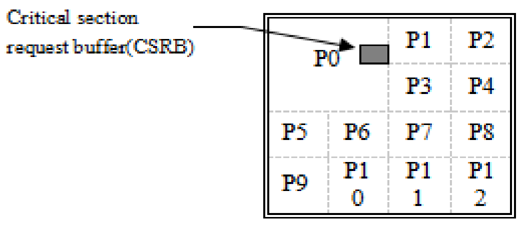


Fig.1. An ACMP structure which used IACS

A. Priority Factor Calculation Method

In order to make the critical sections which recently frequently rejected with a greater priority factor and make better use of the local principle of time.The priority factor Pn is calculated by the formula (1).

$$P_n = P_0 + (Pl_0 - P_0) * RC + R_0 \qquad (1)$$

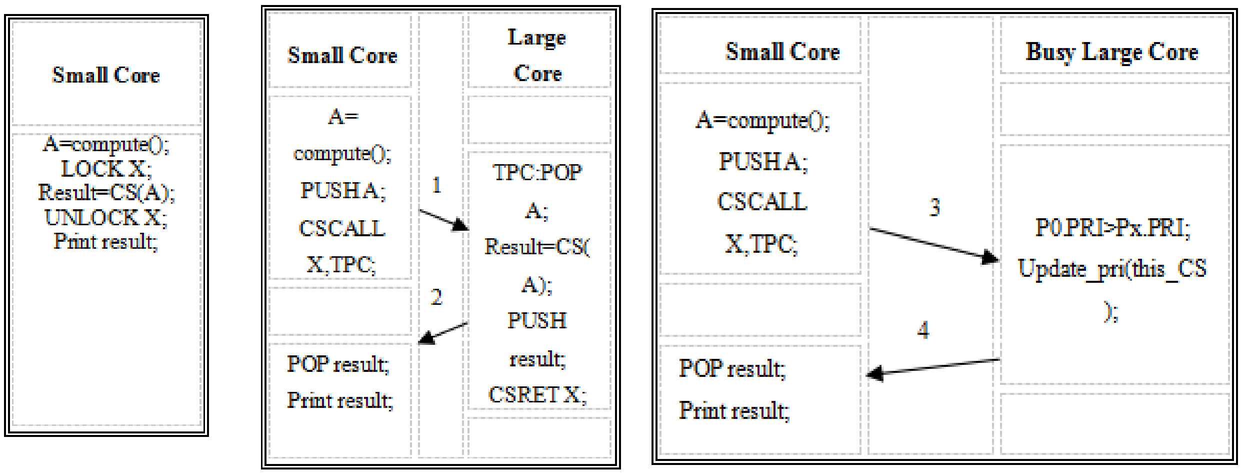
Where Po is the previous priority factor of the critical section，Plo is the priority factor of the critical section which is executing on the high speed core .*RC* is the weight of difference between the executing critical section’s priority and the critical section’s priority.Ro is a fixed constant avoiding priority factor doesn’t change when they are same.

B. IACS Design Ideas

In order to solve the problem of the non-related critical section serialization caused by the ACS method，IACS method is proposed in this paper，The method solves the error of the critical section serialization by answering the question of when the migration happens.

In order to better resolve if migrate threads executing critical section to high speed core.The priority of the critical section is updated by the formula (1) after critical section is rejected.The priority factor guide the migration.

In ACS，when the low speed core executes to the CSCALL,it send CSCALL to P0 and suspend.When P0 receives CSCALL.If P0 does not execute any other critical section，then start executing the critical section until the CSRET instruction is encountered，and then send the CSDONE signal to the original low speed core.Low speed core continue to execute.If the P0 is executing other critical sections，then conduct priority factor judgement，If the current priority factor is low，then migration occurs and wait.When the high speed core finishes the critical section，P0 sends the CSDONE signal to the original low speed core and low speed core continue to execute.Otherwise，P0 send CSRF to the original low speed core to reject.The low speed core continues to execute the critical section and update the corresponding priority factor in the CSRB priority factor table by formula (1).Figure 2 shows the process.(a) is the process of application of IACM before the critical section code execution，(b) is the process of critical section code execution when migration happens in IACM.is the process of critical section code execution when migration not happens in IACM.



(a) (b) (c)

Fig.2.The migration of threads for executing critical

The operation of the digital representation in figure 2, is as follows:

1— CSCALL Request;

Send X,TPC.STACK_PTR,CORE_ID;

2— CSDONE Response;

3— CSCALL Request;

Send X,TPC.STACK_PTR,CORE_ID,PRI;

4— CSRF SIGNAL;

From this we can know that this strategy contains two processes:

1）When the critical section is rejected the current critical section priority factor will be updated by the formula (1) .

2）To determine whether the migration occurs according to the priority factor of the critical section

Locks	Workload	Description	Input set
Coarse	ep	Random number generator	162144 nums
	is	Integer sort	n=64K
	pagemine	Datamining kernel	10Kpages
	puzzle	15-Puzzle game	3*3
	qsort	Quicksort	20K elem.
	sqlite	Sqlite3 database engine	OLTP-simple
	tsp	Traveling salesman prob.	11cities
Fine	iplookup	IP packet routing	2.5K queries
	oltp-1	MySQL server	OLTP-simple
	oltp-2	MySQL server	OLTP-complex
	specjbb	JAVAbusiness benchmark	5 seconds
	webcache	Cooperative web cache	100K queries

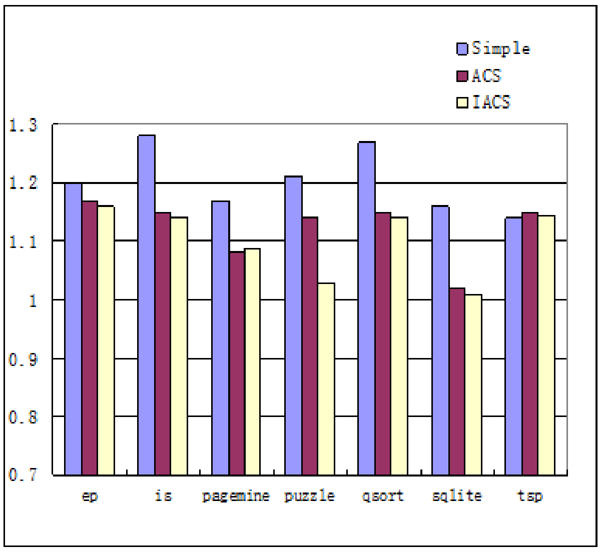
Table 1 The workloads of critical section and their description

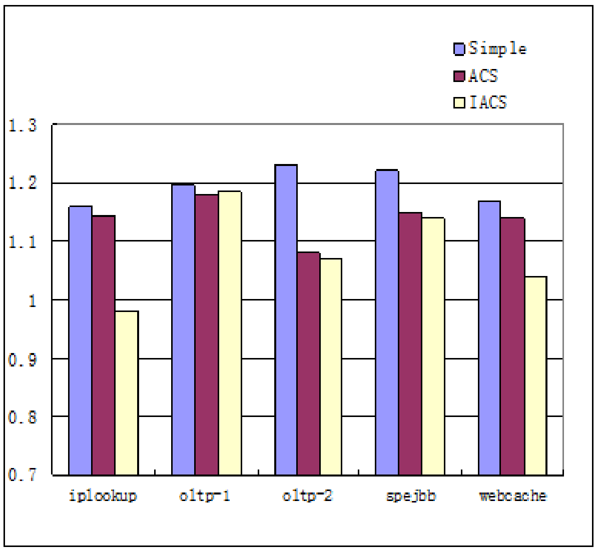
Test results

A. Experimental Environment and Methods

1) experimental Environment:Simulation experiment uses Simics simulating multi-core heterogeneous processor architecture including 1 high-speed cores and 12 low speed core.High speed core support out-of-order execution，main frequency is 3.0GHz，32KB L1 level Cache，1MB L2 level Cache，6MB L3 level Cache.Low speed core does not support out-of-order execution，Main frequency is 2.0GBHz，32KB L1 level Cache，64KB L2 level Cache，3MB L3 level Cache.

2) Test Procedure:In this paper, we mainly evaluate the 12 critical sections of intensive load.The load and its description are shown in Table 1.We define that concentrated load those have 1% instructions executing in critical section.We divide the load into two types：With coarse grain type locks and fine grained locks.We call the load with no more than 10 critical sections as coarse grain size，according to this classification，there are 7 loads using coarse grain locks，the remaining 5 loads use fine grain locks.





(a)Coarse grain load operating conditions (b) Fine grain load operating conditions

Fig.3.The time statistics of the three strategies under different load

B. Experiment and Result Analysis

We compare the simple mapping method, ACS and IACS .The three are based on ACMP.The coefficients of priority formula are set up as follow:RC=0.5，R0=0.2.The coefficients were determined by several experiments.The number of threads is set to the same as the number of cores 12.

·Naive mapping method：The threads are mapped to the core in order of First come First Served，no migration occurred during running time.

·ACS：During the operation, when the critical sectionis encountered,migration happened.

·IACS：An optimization method for ACS proposed in this paper.

Using three different strategies to run the sample program in A-2).The time of completion of statistics is shown in Figure 3.Figure 3 (a) represents the 7 load using coarse grain locks run time statistics，Figure 3 (b) represents the 5 load using fine grain locks run time statistics.From the graph,ACS and IACS in each program are better than simple mapping method.Because the IACS does not affect the program that are rarely encountered serialization.We focus only on and evaluate three case programs that will encounter wrong serialization.：puzzle、iplook and webcache.Figure 3 shows the Standardized execution time of simple mapping method、ACS and IACS executing puzzle、iplook and webcache.Due to the iplookup and webcache case program will encounter more wrong serialization.So when using the IACS algorithm，Their performance was promoted for 16% and 9%.

In general，For all 12 loads，the average performance of the IACS method is improved by 17% compared with the ACS method.We conclude that IACS can successfully reduce the execution time of the critical section that are encountered in the wrong serialization and do not affect the execution of the program that does not encounter wrong serialization.

Conclusion

In this paper, we mainly aim at the dynamic scheduling problem of the thread group of the critical section density and proposed IACS algorithm.The algorithm tracks the wrong serialization behavior of all critical sections,give higher priority to the critical section of frequent rejection by high speed and provides the basis for the thread scheduling from low speed core to the high speed core,shorten the execution time of the critical area， improve the overall performance of the system.At the same time the wrong seavoid the problem of ACS method , which is likely to cause the serial of the irrelevant critical section.Simulation results show that the IACS algorithm is shorter than the ACS method and the performance is better.The deficiency of the algorithm is that the algorithm is more complex and the migration overhead of the thread is more limited to the load type than the ACS.The future direction is to research how to better balance algorithm overhead and system performance enhancement， Resource allocation strategies are also listed in our future work plan.

References

[1] R. Kumar, K. I. Farkas and N. P. Jouppi et al., “Single-ISA Heterogeneous Multi-Core Architectures: The Potential for ProcessorPower Reduction,” International Symposium on Microarchitec-ture, 2003.

[2] T. Sherwood, S.Sair and B.Calder, Phase tracking and predic-tion, Proceedings of 30th Annual International Symposium on Computer Architecture, pp. 336- 347, 2003.

[3] Chen Rui-zhong, QI De-yu,LIN Wei-wei,et al,A virtual machine for asymmetric multi-core processors integrated scheduling algorithm[J], Chinese Journal of Computers, 2014,07:1466-1477.

[4] Nie Peng-cheng,DUAN Zhen-hua,TIAN Meng,et al, Performance of asymmetric adaptive scheduling of multi-core processors[J], Chinese Journal of Computers, 2013,04:773-781.

[5] Yang Peng-fei,WANG Quan, On-chip network heterogeneous polynuclear system task scheduling and mapping[J], Journal of Xi'an Jiaotong University,2015,06:72-76+125.

[6] M. A. Suleman, O. Mutlu, M. K. Qureshi, and Y. N. Patt, “Accelerating critical section execution with asymmetric multi-core architectures,” in Proc. 14th Int. Conf. on Architectural Support for Programming Languages and Operating Syst. (ASPLOS’09), 2009, pp. 253–264, ser.