GLIMPSE -Journal of Computer Science •Vol. 1(1) , JANUARY-JUNE 2022, pp. 11-12

ASYMMETRIC MULTI-CORE ARCHITECTURES

Santosh Kumar Upadhyay
Assistant Professor, Ajay Kumar Garg Engineering College,Ghaziabad, UP, India
ersk2006@gmail.com

Abstract: **For the majority of applications, multi-core architectures are built to give a reasonable degree of performance per unit power. To get the good result from chip multiprocessors (CMPs), we need to divide our program into threads that run on several cores at the same time. This concurrent execution may arise critical section problem. In this paper, the brief overview of asymmetric multi-core processors is given to accelerate the critical section execution.**

Keywords: **CMP, Critical Section, Threads, Parallel processing, Multi-core.**

I. INTRODUCTION

In Multiprocessing many processors work at the same time in computer system. The central processing unit is the computational unit that runs programs and applications. The CPU is the arithmetic and logic engine that executes applications and programs. More than one user program can run at the same time using several CPUs. An ongoing job can be rescheduled from one CPU to another since all of the CPUs share the same user-mode instruction set. An asymmetric multi-core processor, on the other hand, contains numerous cores on a same chip that may be of different architectures.

A symmetric multicore processors are made up of cores that share the same instruction set architecture but differ in efficiency, complication, and energy usage [1], [2]. An Asymmetric multicore processor may consist of many slow, tiny, and simple cores as well as a small number of fast, massive, and complicated cores. A more power saving choice to symmetric multicore CPUs has been suggested: AMPs. Because many threads cannot edit shared data at the same time, access to shared data is wrapped within crucial segments.

A crucial part is only executed by one thread at a time; other threads wishing to run the same crucial part must delay. Threads can be serialized at critical areas, lowering flexibility and efficiency (that is, the threads count at which efficiency stabilizes). This efficiency degrades can be mitigated by reducing the running time within crucial portions.

A critical section is only executed by one thread at a time; additional threads that want to execute the same critical part must wait. Threads can be serialized in critical areas, lowering scalability and effectiveness (that is, the threads count at which performance saturates). By decreasing the running time inside crucial part the performance loss can be minimized.

CMP cores can be symmetric (SCMP) or asymmetric (Asymmetric CMP) (ACMP). Varied workloads demand different CPU resources for superior efficiency, as is widely known. Some loads are load-store intensive, while others are integer ALU intensive, floating-point (FP) intensive, memory bus intensive, or a mix of the two. [3]. An ACMP has a higher chance of delivering results while using fewer resources.

II. PROBLEM AND RELATED WORK

There are two pieces to a multithreaded application. Amdahl's bottleneck is represented by the serial component. A parallel portion is one in which multiple threads run at the same time. At any one-time, single thread can run a critical segment. Other threads that do not require execution of that crucial segment can progress while it is being executed. [4].

All previous research has been focused on improving the program's implementation in critical sections. When a process operates in the critical section, it makes all three portions of the kernel busy, namely the beginning, end, and parallel parts, which are all run by numerous threads. As a result, even if just one process requests, the entire system becomes busy.

The optimization part was previously addressed by following actions:

- Improving the shared data locality and locks.
- Concealing the delay of critical segments.
- Asymmetric chip multiprocessors
- Remote procedure calls

III. PROPOSED SOLUTION

Asymmetric multi-core processors have numerous cores on a same chip, although they may be of distinct architectures. These architectures can also be used to speed up the critical section. IBM's Cell processor is a nice example currently on the market. An ACMP is employed in a video game named as the Sony PlayStation 3. The Cell is equipped with nine processing unit cores, eight data-processing cores and

one general-purpose processing unit. The Power Processor Element (PPE), a single multifunctional core, manages interactions in the other cores and assigns computational workloads to another cores for computing. Synergistic Procedural Cores are the remaining eight cores and are optimized for good floating-point throughput, particularly when performing vector operations.

The basic architecture [4] is proposed for speeding the critical part using ACMPs. There are 13 processors in this architecture. The serial and important sections of the application run on the high-performance core, while the remaining parallel sections run on the tiny cores.

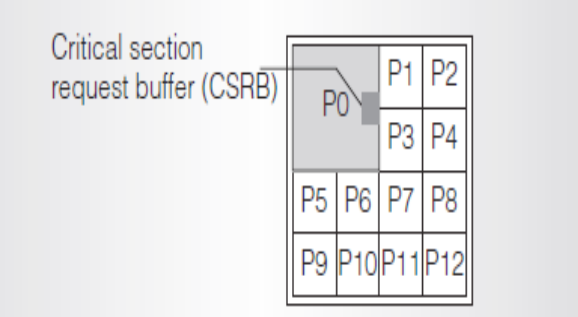

Figure 1. An Asymmetric chip architecture consists of one high performance core and several smaller cores.

Figure 1 depicts an ACS design developed on an ACMP with one major core (P0) and twelve mini cores (P1 to P12). P0 is dedicated to the execution of important sections by ACS, which executes parallel threads on tiny cores (as well as serial program portions). The critical section execution requests from the tiny cores are buffered by a critical section request buffer (CSRB) in P0. Critical section return (CSRET) and critical section execution call (CSCALL) are the two new accelerating critical section instructions that are entered at the start and final point of a crucial region, respectively.

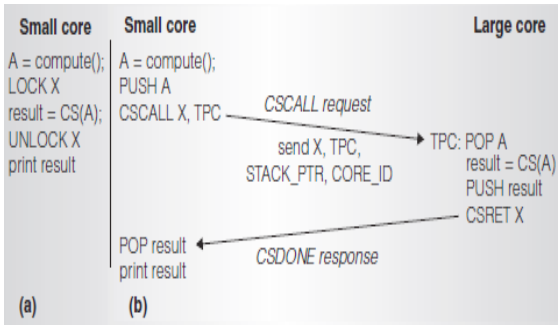

Figure 2. Source code and its execution: baseline (a) and ACS (b).

The comparison of ACS to a typical system is shown in Figure 2. In classical locking (Figure 2a), when a micro core discovers a crucial part, it achieves the lock safeguarding the crucial segments, runs the crucial part, and releases the lock. When a tiny core in ACS (Figure 2b) performs a CSCALL, it sends the CSCALL to P0 and waits until it gets a reply. When CSCALLs come, P0 buffers them in the CSRB. P0 starts running the designated crucial section as quickly as possible, then switches to regular processing until it finds a CSRET instruction, that shows that the crucial section has finished. When P0 runs the CSRET command, it transmits a critical section done (CSDON) signal. When P0 executes the CSRET instruction, it sends a critical section done (CSDONE) signal to the asking small core. After getting this indication, the small core resumes normal operation.

Shorter jobs will run on small cores, whereas larger jobs will run on a large core, owing to its design. The high-performance core will be in responsible of scheduling.

IV. CONCLUSION

In this paper, asymmetric multicore processor is discussed with its present implementation in IBM's cell processor which is used in the Sony PlayStation 3 video game console which consists of 9 processors. And a valuable model is discussed which accelerates the critical section execution. Using this model, the hardware is changed in such a manner to accelerate the critical section, before this the optimizations are done on the basis of programming. But here the asymmetric multicores are designed in such manner to make the executions much faster.

Still there are lots more methods to improve the performance and accelerate the critical section. We can also implement both the software as well as hardware aspects to improve our results by making a bit change in use of locks, caching mechanism, scheduling, etc.

V. REFERENCES

[1] R. Kumar et al. *Single-ISA Heterogeneous Multicore Architectures for Multithreaded Workload Performance,* ISCA, 2004.

[2] Jian Li and Jose F. Martinez. "*Dynamic Power-Performance Adaptation of Parallel Computation on Chip Multiprocessos, in High-Performance Computer Architecture"*, 2006.

[3] Anup Das, Rance Rodrigues, Israel Koren and Sandip Kundu, "A Study on Performance Benefits of Core Morphing in an Asymmetric Multicore Processor" in IEEE 2010.

[4] M. A. Suleman, O. Mutlu, M. K. Qureshi, and Y. N. Patt. "*Accelerating Critical Section Execution with Asymmetric Multicore Architectures*" in IEEE, 2010.

ABOUT THE AUTHOR

Santosh Kumar Upadhyay is an Assistant Professor in the CSE deptt. at AKGEC, Ghaziabad, UP, India since 2018. He has received Bachelor's degree in computer Science and Engineering from UPTU in 2005. and Master with Honors in Information Technology.

He is silver medalist in Master of Technology in Tezpur University. The major fields of study is network security, data mining, Machine learning.