社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2003－MPS－45 (10)
2003／6／24

論理スレッド番号により管理されるキャッシュを持つマルチスレッドプロセッサの性能評価

Hironori Nakajo ,† Masanori Yamato ,† Shoji Kawahara ,††
Norito Kato ,† Koichi Sasada ,† Mikiko Sato †
and Mitaro Namiki †

Currently, multi-threaded architectures such as chip multi-processor and SMT (Simultaneous Multi-threading), which exploit TLP in addition to ILP, are in a hot topic. In such architecture, however, simultaneously executed threads cause conflicts in cache entries among threads, thus it may degrade efficiency of cache. In this paper, we propose an LTN based replacement strategy that utilizes thread number: Logical Thread Number (LTN) managed by OS in order to control a thread to be replaced in cache entry. We have evaluated our proposed strategy by simulator MUTHASI. Since it is not necessary to add so much hardware resources for the LTN replacement strategy, it is expected that the LTN based replacement strategy brings high hit ratio without expansion of chip area.

1. Introduction

1.1 Multiprocessing and a chip multiprocessor

It is getting common that a symmetric multiprocessor (SMP) with two, for or more processors is utilized as a server machine in a class of high-end PCs. Moreover, some motherboards which holds dual processors are getting available with inexpensive costs, thus ordinary users can utilize the ability of parallel processing.

However, the more the number of processors increases, the more electric power is needed. If a license of an operating system is needed for a single processor, the cost of multiprocessing system cannot be ignored.

While a process rule of semiconductor is getting more small, some processor vendors have been trying to implement general purpose chip multiprocessors which hold multiple processors in a single chip.

Chip multiprocessor technology has been already applied to an application specific usage such as network processing. Furthermore, in research levels, there have been many researches on chip multiprocessors such as Hydra of Stanford[1], Power4 of IBM[2] and so on. However, only coupling multiple processor cores into a single chip cannot achieve higher performance, therefore some integration are needed such as memory or input/output technologies.

† 東京農工大学工学部情報コミュニケーション工学科
Department of Computer, Information and Communication Sciences, Tokyo University of Agriculture and Technology
†† NEC シリコンシステム研究所
NEC Silicon Systems Research Laboratories

1.2 Emergence of Simultaneous Multithreading (SMT)

In the situation mentioned above, Simultaneous Multi-threading (SMT) architecture which can achieve higher performance even with a single processor has emerged recently.

Multi-threaded architecture, which was first introduced in release of HEP which was designed by Burton Smith, aims latency hiding that switches threads executing an instruction which causes a long access latency. Based on multi-threading technology, SMT, which combines ILP and TLP and executes multiple threads efficiently has been proposed as a new technology which utilizes unused resources maximally.

A very long instruction word (VLIW) architecture is currently available as a multiprocessing technology which utilizes multiple resources at a instruction level. However, there are significant common problems in compiler development and different instruction sets are needed. On the other hand, SMT utilizes an existing instruction set though supports of an operating system and a compiler are necessary,

Recently, Intel has released Hyper Threading[3] which is developed in a project called Jackson. Hyper Threading, dispatch two threads to a single processor, is a first processor which implements an SMT technology. From the point of view of an operating system, though a system is equipped with just a single processor, two processors were seemed to be executed simultaneously. However, performance gain of Hyper Threading is expected up to 10 – 30 percent, because each internal bus, cache memory, operation unit and execution unit is just equipped in a processor solely.

In such SMT architecture, however, simul-

taneously executed threads cause conflicts in cache entries among threads, thus it may degrade efficiency of cache.

Thus, we propose a cache replacement strategy which utilizes thread number: Logical Thread Number (LTN) managed by OS in order to control a thread to be replaced in cache entry.

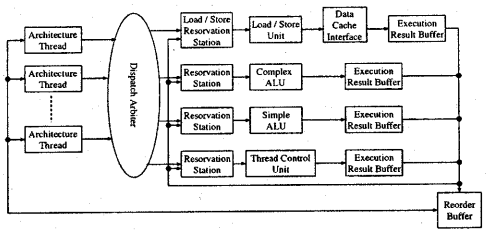
We have evaluated our proposed strategy by simulator MUTHASI (MUlti-THreaded Architecture SImulator). Since it is not necessary to add so much hardware resources for the LTN based replacement strategy, it is expected that the LTN based replacement strategy brings high hit ratio without expansion of chip area. We have been also developing a chip multiprocessor based on an SMT architecture called OChiMuS (On-Chip Multi-SMT).

2. Architecture of a target SMT processor

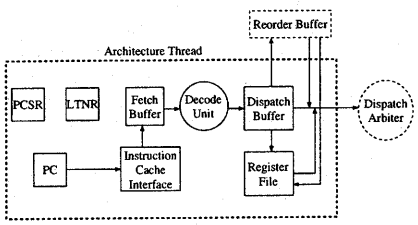
2.1 Concept of a thread

A thread is defined as a series of instructions and a process is a set of one or more threads which share the same memory space in the same process.

We propose a processor architecture supported with virtualization of a thread management in which system software doesn't need to manage assignment status of a Program Counter (PC) or a logical thread and free PC's.



(a)Overall targeted architecture



(b)Architecture thread oriented part

Fig. 1 Configuration of a targeted SMT processing element

For this reason, we also propose a management method which cooperates with an operating system with dividing threads into logical threads and architecture threads. A logical thread is a thread managed by system software and an architecture thread is a being executed thread allocated into a PC.

2.2 Processor configuration

Figure 1 shows a configuration of a targeted SMT processor. In this architecture, each architecture holds resources as follows.

- Program Counter (PC)
- Program Counter Status Register (PCSR)
- Logical Thread Number Register (LTNR)
- Register File (RegisterFile)

Program Counter Status Register (PCSR) holds a status of being executed thread. Logical Thread Number (LTN) is stored in a Logical Thread Number Register (LTNR) and referred in a thread control instruction. Each PC has its own instruction cache and data cache is shared among all PC's.

2.3 Thread control

A thread control instruction specifies not a PC but an LTN. For example, in the case of thread allocation, hardware searches a free PC which is not assigned with a logical thread then assigns the thread to the PC when exists. When there is no free PC, the instruction fails. In other cases of thread instructions, a PC which is assigned with LTN is searched and the thread control instruction is issued if it exists, otherwise it fails.

By management of LTN by hardware, system software doesn't concern whether a PC is allocated with a thread or which thread is allocated to which PC.

On the contrary, without this mechanism, information of which logical thread is allocated to which PC should be kept in memory and be referred in each time of management of a thread, thus it causes a significant overhead of a thread management. Consequently, a thread management with using LTN seems to be effective in a multi-threaded processor especially for fine grained applications.

3. LTN based replacement for cache of an SMT architecture

As a problem caused by sharing data cache among threads in a processing element (PE), when multiple threads are executed in a PE, accessing the same entries frequently occurs in a multi-threaded architecture. Thus cache miss caused by conflicts increases, which brings significant cache miss penalty in accessing data. Therefore, in a multi-threaded processor, effects of cache would degrade by cache replacement which is seldom in a single-threaded processor.

In order to overcome the problem, increasing capacity of cache or associativity may effective, however, it would brings increase of a size of a chip and access latency, thus performance gain cannot be expected against additional hardware

resources.

We propose a cache replacement strategy using Logical Thread Number (LTN). By utilizing LTN when specifying a way to be replaced, since a way to be replaced is fixed dependent of a PC which is allocated to a thread, the way is not influenced by a thread context. We call the proposed strategy LTN based replacement. Figure 2 shows the concept of an LTN based replacement.

In the LNT based replacement, when the number of ways which can be replaced is N, it is called as N set LNT based replacement. For example, in a 4 way set-associative cache, when one bit is used for modulo of LTN and the possible numbers of ways to be replaced for one LTN is two, we call it 2 set LTN based replacement.

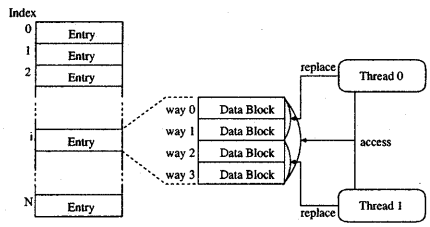


Fig. 2 LTN based replacement strategy

4. Simulation based evaluation of cache

We have evaluated cache memory of a multi-threaded processor with a multi-threaded architecture simulator called MUTHASI (MUlti-THread Architecture SImulator) comparing a proposed LTN replacement with LRU.

MUTHASI is an instruction level simulator, which based on MIPS R4000 architecture with thread control instructions, that can exactly simulate each stage of pipeline in a multi-threaded processor and can easily change the number of PCs.

We have adopted an own developed thread library called MULiTh[4] which is applied to thread instructions based on a POSIX thread.

4.1 Evaluation of 2 way set-associative

Figure 3 shows a result of a multiplication of a 64×64matrix. In the graph, the vertical and horizontal axes mean the number of execution cycles and the size of cache in some numbers of PCs respectively.

From the result, LTN replacement is inferior to LRU, however, except for the single PC with cache sized of 16KB, performance degradation is around 1 % in this size of matrix.

Next, a result of multiplication of matrix sized 256×256 is shown in Figure 4. Totally, it is found that the LTN replacement is effective except for the case of the 32KB sized cache. Especially, in the cases of the size with 64KB and 128KB, performance gain is large.

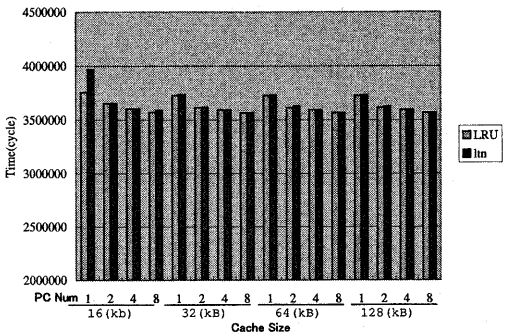


Fig. 3 64×64 matrix multiplication of 2 way

Amount of data accessed for a matrix is 256KB($=4byte \times 256 \times 256$) which exceeds the capacity of cache.

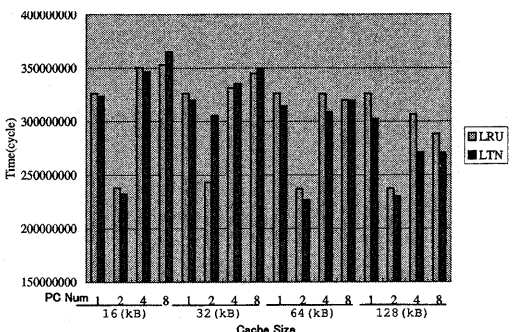


Fig. 4 256×256 matrix multiplication of 2 way

4.2 Evaluation of 4 way set-associative

5 shows the result of a matrix multiplication with 4 way set-associative cache.

In the case of cache capacity with 16KB, performance degradation in the case of one and two PCs, however, the performance is almost same in other cases. Moreover, there is little difference between 2 set and 1 set LTN replacements.

Figure 6 shows a result of multiplication of matrix sized 256×256.

From the results, LTN based replacement is tend to be superior to LRU one. As the case of 2 way set-associative, the more the amount of accessed data is, the more effective the LTN based replacement.

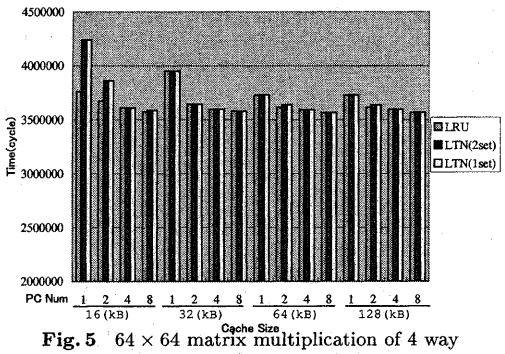


Fig. 5 64 × 64 matrix multiplication of 4 way

Furthermore, single set LTN based replacement is faster than 2 set LTN based one. Especially, in 4 PCs with cache sized 128KB, though 2 set LTN based replacement is 12% faster than LRU, 1 set LTN based replacement is 26% faster than LRU one.

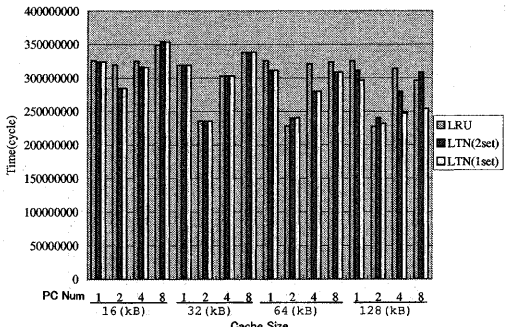


Fig. 6 256 × 256 matrix multiplication of 4 way

5. Implementation of LTN based replacement

In order to implement LTN based replacement, following additional hardware resources are needed.

- A bus for transferring lower one or two bits of LTN modulo to a load store unit.
- A buffer in each entry for storing LTN modulo in a dispatch queue and load store reservation station.
- Signal lines for LTN modulo in a data request bus.

Especially, no tag space is needed for LTN based replacement, it is possible to implement in low costs because additional hardware for above resources is sufficiently small comparing to an area of a processor core.

Though there are 64 bits bus, 32 bits for address and the other 32 bits for data, for memory accessing, additional bus from one to 3 bits are significantly small. Consequently, additional bus is less than 1% in a whole processor core, thus it is sufficiently small for performance gain.

6. Conclusion and Future work

In this paper we have proposed a new cache replacement strategy called LTN based replacement. We have evaluated the strategy by instruction level simulator MUTHASI and found that the larger data size increases the more speed up is gained by the strategy with less hardware costs against LRU.

We have been designing an SMT procossor with the cache replacement strategy. Moreover we have been studying a chip multiprocessor called OChiMuS (On-Chip Multi-SMT).

Requirements for implementation of OChiMuS from the viewpoints of a system architecture are shown as follows.

- Integration of memory architecture which is significantly important in an SMT architecture.
- Fast synchronization which makes use of advantages of an SMT architecture.
- Specification of roles of thread management between an operating system[5] and an architecture.

Concerning a compiler, it is efficient to utilize various parallelizing technologies which are built in existing SMP.

In order to aim a general purpose and speedup of threads in multitasking, a dedicated operating system, a specialized compiler and also programming languages should be discussed for the future high performance computing.

References

1) L. Hammond, B. Hubbert , M. Siu, M. Prabhu , M. Chen , and K. Olukotun: The Stanford Hydra CMP, IEEE MICRO Magazine, 2000, and presented at Hot Chips 11, 1999
2) J. Kahle: Power4: A Dual-CPU Processor Chip, Microprocessor Forum'99, 1999.
3) http://www.intel.com/technology/hyperthread/
4) "Implementation and Evaluation of a Thread Library for Multithreaded Architecture"
K. Sasada, M. Sato, S. Kawahara, N. Kato, M. Yamato, H. Nakajo and M. Namiki,
Proc. of International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2003) (2003)
5) "A Process and Thread Management of the Operating System "Future"for On Chip Multithreaded Architecture"
M. Sato, K. Sasada, S. Kawahara, N. Kato, M. Yamato, H. Nakajo and M. Namiki,
Proc. of International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2003) (2003)