Analysis of the Impact of Simultaneous Multithreading on Cache Interference

Byung In Moon
Yonsei University
Seoul, Korea
bihmoon@soc.yonsei.ac.kr

Ilgu Yun
Yonsei University
Seoul, Korea
iyun@yonsei.ac.kr

Hongil Yun
Yonsei University
Seoul, Korea
hyoon@yonsei.ac.kr

Sungho Kang
Yonsei University
Seoul, Korea
shkang@yonsei.ac.kr

Abstract – *Simultaneous multithreading attracts attention to many microprocessor designers, since it seems to overcome the architectural performance limit of the superscalar processor. Although simultaneous multithreading improves processor utilization through dynamic resource sharing among threads, it can increase cache interference and degrade overall performance. This paper identifies the problem of inter-thread cache interference on simultaneous multithreading and analyzes the impact that variation of cache configuration parameters has on inter-thread cache interference and overall performance. Our results show that the size of caches has a big impact on both inter-thread and intra-thread cache interference. On the other hand, increasing the number of cache ways is beneficial only to inter-thread cache interference. Overall, the variation of the size and set associativity of caches have a significant impact on processor throughput, whereas the change of the block size of caches has little impact.*

Keywords: Simultaneous multithreading, cache, inter-thread cache interference, intra-thread cache interference.

1 Introduction

The superscalar processor reaches its architectural performance limit. For this reason, multithreading, especially simultaneous multithreading (SMT) [1] draws special attention to many microprocessor designers which have been seeking to improve the processor architecture for higher performance. Multithreading hides latency problems such as memory access latency by increasing parallelism through thread-level parallelism (TLP), thus dramatically increasing processor utilization and significantly improving instruction throughput. Multithreading is divided into three categories: coarse multithreading (CMT), fine multithreading (FMT), and SMT [2]. A key feature of multithreading is sharing processor resources among threads. In CMT and FMT processors, at most one thread can occupy a pipeline stage at one time, so inter-thread resource sharing is limited to an interleaving of instructions from different threads. On the other hand, in the SMT architecture, the processor pipeline resources may be shared horizontally as well as vertically, so that SMT can convert TLP to instruction-level parallelism (ILP) [3]. Through this parallelism converting, the SMT architecture issues and executes instructions from multiple threads each cycle, increasing processor throughput notably.

Although multithreading improves processor performance by exploiting TLP, it has a tendency to increase the number of conflict cache misses. And increased cache miss rates by multithreading can hurt processor performance. In CMT, context switching between threads may evict useful cache blocks from descheduled threads. In SMT, where thread execution is interleaved at a much finer granularity, cache accesses from one thread may replace active cache blocks from other threads with the accessed blocks. This inter-thread cache interference may offset the performance improvement by multithreading. This paper analyzes cache interference on SMT processors with various cache configurations. We categorize cache misses in the SMT architecture as first-reference misses, inter-thread conflict misses, and intra-thread conflict misses [3], and we measure these three types of cache miss rates on SMT processors with a variety of cache configurations. This paper also presents cache configuration factors that significantly affect inter-thread cache interference on SMT processors.

The problem of inter-thread cache interference in multithreading architectures has been studied by a number of researchers. Thekkath and Eggers [4] analyzed the impact of thread placement algorithms on the cache interference of multithreaded multiprocessor architectures. Lo et al. [3] measured inter-thread conflict misses on SMT processors through simulations, during which they varied the number of supported threads but fixed the cache configuration. Lo et al. [5] examined the effects of cache interference on SMT processors for database workloads. These previous works do not give full analysis of the impact that the cache configuration has on inter-thread cache interference in the SMT architecture. This paper is, to our knowledge, the first to analyze inter-thread cache interference on SMT processors with various cache configurations.

2 Simulation methodology

This section presents an overview of the base processor architecture that we model in detail for simulations, and gives some explanations for the modelling and simulation methods.

2.1 Base architecture

In [6] and [7], an in-order SMT architecture was introduced that is used as the base processor architecture of this paper. We summarize this architecture in this section, but the interested reader is referred to [6] and [7] for more details.

Our SMT architecture is pipelined into 8 stages: *select* (S), *fetch* (F), *decode* (D), *issue* (I), *read* (R), *execute* (E), *memory* (M), and *write* (W). In the S stage, which is an extra stage for supporting multithreading, the thread selector picks up threads from which instructions are to be fetched by the fetch unit in the next cycle. In the D stage, the decode unit decodes instructions in the fetch order. The issue unit issues decoded instructions to the functional units. Issued instructions obtain their operands through bypassing, or from the register file. Then they are executed in their assigned functional units. Exception and branch misprediction are checked in the M stage. In the W stage, writes to the register file are performed in the order within each thread, and instruction execution is completed.

Table 1. Base configuration parameter values.

Configuration parameter	Parameter value
Issue width (IW)	4 or 8 instructions per cycle
Fetch width	(1 × IW) instructions per cycle
Instruction fetch queue (IFQ)	(8 × IW) entries
Instruction issue queue (IIQ)	(8 × IW) entries
Register file	(2 × IW) read ports, (1 × IW) write ports
Functional units	(1 × IW) integer ALUs, (1/2 × IW) integer multipliers, (1/2 × IW) load/store units
Latencies	1 cycle except integer multiplication (2 cycles)
BTB	(32 × IW) entries, branch prediction using 2-bit saturation counters
Instruction TLB	64 entries
Data TLB	64 entries
Instruction and data caches	(1/2 × IW) ports, 1-cycle hit latency, 10-cycle miss penalty, random replacement
Fetch policy	Give higher priority to those threads with fewer instructions in the IFQ and IIQ
Issue policy	Give higher priority to those instructions closer to the head of the IIQ

2.2 Modelling and simulation

We measure inter-thread cache interference using an execution-driven, cycle-based simulator modelling in detail our SMT architecture. The simulator adopts the ARM architecture [8] as its instruction set architecture (ISA), and its base configuration parameter values are listed in Table 1.

Our workload for simulation is from the three SPEC CPU2000 benchmarks: *mcf*, *twolf*, and *vortex* [9], and from the two example programs of the ADS (ARM Developer Suite): *Dhrystone* and *sorts* [10]. Threads of the simulator model execute the five programs in different orders to prevent abnormal inter-thread cache interference, which can occur when multiple threads execute the same program simultaneously. During the simulations using this workload, we vary three configuration parameters of instruction and data caches: *cache size*, *set associativity*, and *block size*, and measure cache miss rates and instructions per cycle (IPC) for each cache configuration. For convenience, the instruction and data caches have the same parameter values.

3 Simulation results

We carried out simulations for identifying the problem of inter-thread cache interference on SMT. As shown in Fig. 1, as the number of threads increases, the number of inter-thread conflict misses of the instruction cache rises, from 4.7% (2 threads) to 10.4% (4 threads) to 12.5% (8 threads) to 18.9% (16 threads) of total instruction cache accesses. The data cache shows the similar results for inter-thread cache interference. This increase of inter-thread cache interference in the SMT architecture has a

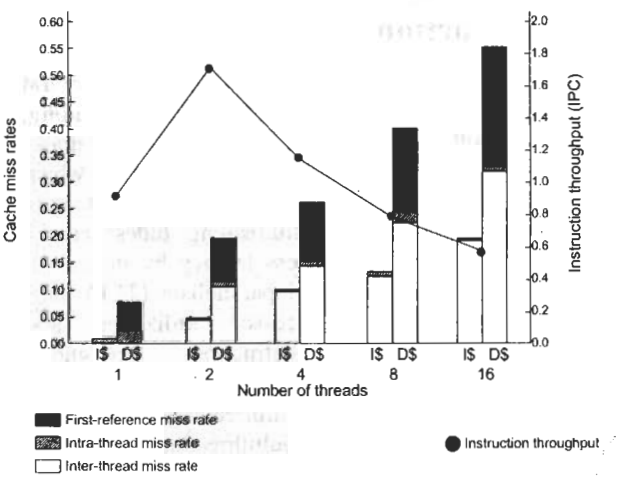

Fig. 1. Cache miss rates and instruction throughput for 8-issue processors (using 8-KB direct-mapped caches whose block size is 32 bytes).

Table 2. Variation of cache miss rates and instruction throughput depending on cache size.

Processor configuration	Cache size	Cache miss rate		Inter-thread conflict miss (miss rate/ ratio to total miss rate)		Intra-thread conflict miss (miss rate/ ratio to total miss rate)		Instruction throughput (IPC)
		I$	D$	I$	D$	I$	D$	
4-issue, 4-thread, 4-way set associative, block size fo 64 bytes	2 KB	0.0903	0.2015	0.0690/76%	0.0818/41%	0.0206/23$	0.0223/11%	I.3210
	4 KB	0.0601	0.1284	0.0457/76%	0.0433/34%	0.0137/22%	0.0124/10%	1.9187
	8 KB	0.0253	0.0670	0.0191/75%	0.0120/18%	0.0056/22%	0.0036/5%	2.5225
	16 KB	0.0020	0.0548	0.0014/70%	0.0049/9%	0.0004/20%	0.0018/3%	2.6839
	32 KB	0.0004	0.0515	0.0002/56%	0.0028/5%	0.0001/21%	0.0013/3%	2.6958
8-issue, 8-thread, 8-way set associative, block size of 64 bytes	4 KB	0.0902	0.3050	0.0802/89%	0.0896/29%	0.0094/10%	0.0132/4%	1.2488
	8 KB	0.0528	0.1981	0.0469/89%	0.0460/23%	0.0052/10%	0.0082/4%	2.2247
	16 KB	0.0280	0.1400	0.0244/87%	0.0233/17%	0.0032/11%	0.0048/3%	4.2015
	32 KB	0.0069	0.1120	0.0057/83%	0.0114/10%	0.0007/10%	0.0034/3%	5.2828
	64 KB	0.0014	0.1000	0.0011/78%	0.0062/6%	0.0001/9%	0.0026/3%	5.3925

tendency to degrade overall performance. The performance with 1 thread is better than that with 2 threads. This is because the benefit by multithreading outweighs the increased inter-thread cache interference. To the contrary, as the number of threads increases in the range over 2 threads, the instruction throughput gets lower, since, in this range, the parallelism increased by supporting more threads is more than offset by the increased inter-thread cache interference.

We found in Table 2 that as the size of caches increases with the set associativity and block size fixed, inter-thread and intra-thread conflict miss rates drop, and thus instruction throughput improves. This is because larger caches provide the more capacity to accommodate the working sets. For our workload, 16-KB and 32-KB caches are appropriate for 4 threads and 8 threads, respectively.

Table 3 shows that as the number of cache ways increases, inter-thread conflict miss rates fall, and thus total miss rates decrease and processor throughput improves. However, the variation of set associativity has a limited impact on intra-thread conflict misses unlike that of cache size. This result indicates that the number of cache ways is a very important parameter to the inter-thread cache interference and processor performance of the SMT architecture. We further found that increasing the number of cache ways to more than the number of threads is little beneficial to inter-thread conflict miss rates and instruction throughput. So we recommend 4-way and 8-way caches for 4 threads and 8 threads, respectively.

The large block size of caches is beneficial to sequential memory accesses, but is not favorable to random memory accesses, since the smaller number of blocks is provided if the block size becomes larger with the cache size fixed. So the instruction cache benefits from increasing the block size, but the data cache does not, as shown in Table 4. Overall, the block size of caches has little impact on the inter-thread cache interference and processor performance of our SMT architecture.

4 Conclusions

By converting TLP to ILP, the SMT architecture issues and executes instructions from multiple threads each cycle,

Table 3. Variation of cache miss rates and instruction throughput depending on set associatvity.

Processor configuration	Set associativity	Cache miss rate		Inter-thread conflict miss (miss rate/ ratio to total miss rate)		Intra-thread conflict miss (miss rate/ ratio to total miss rate)		Instruction throughput (IPC)
		I$	D$	I$	D$	I$	D$	
4-issue, 4-thread, cache size of 8 KB, block size of 64 bytes	1-way	0.0445	0.2035	0.0402/91%	0.0861/42%	0.0041/9%	0.0106/5%	1.7897
	2-way	0.0359	0.1217	0.0302/84%	0.0414/34%	0.0054/15%	0.0044/4%	2.2252
	4-way	0.0253	0.0670	0.0191/75%	0.0120/18%	0.0056/22%	0.0036/5%	2.5225
	8-way	0.0190	0.0656	0.0141/74%	0.0105/16%	0.0049/26%	0.0037/6%	2.5677
	16-way	0.0174	0.0612	0.0126/72%	0.0082/13%	0.0048/27%	0.0035/6%	2.5917
8-issue, 8-thread, cache size of 16 KB, block size of 64 bytes	2-way	0.0442	0.2112	0.0421/95%	0.0508/24%	0.0019/4%	0.0060/3%	2.2998
	4-way	0.0350	0.1708	0.0319/91%	0.0354/21%	0.0028/8%	0.0052/3%	3.2805
	8-way	0.0280	0.1400	0.0244/87%	0.0233/17%	0.0032/11%	0.0048/3%	4.2015
	16-way	0.0240	0.1378	0.0207/86%	0.0219/16%	0.0032/13%	0.0050/4%	4.4539
	32-way	0.0235	0.1315	0.0200/85%	0.0184/14%	0.0034/14%	0.0048/4%	4.6147

Table 4. Variation of cache miss rates and instruction throughput depending on block size.

Processor configuration	Block size	Cache miss rate		Inter-thread conflict miss (miss rate/ ratio to total miss rate)		Intra-thread conflict miss (miss rate/ ratio to total miss rate)		Instruction throughput (IPC)
		I$	D$	I$	D$	I$	D$	
4-issue, 4-thread, cache size of 8 KB, 4-way set associative	8 bytes	0.0279	0.0626	0.0209/75%	0.0106/17%	0.0064/23%	0.0044/7%	2.4910
	16 bytes	0.0269	0.0639	0.0202/75%	0.0121/19%	0.0059/22%	0.0032/5%	2.5037
	32 bytes	0.0260	0.0645	0.0192/74%	0.0116/18%	0.0060/23%	0.0039/6%	2.5108
	64 bytes	0.0253	0.0670	0.0191/75%	0.0120/18%	0.0056/22%	0.0036/5%	2.5225
8-issue, 8-thread, cache size of 16 KB, 8-way set associative	8 bytes	0.0307	0.1416	0.0264/86%	0.0255/18%	0.0034/11%	0.0057/4%	4.1395
	16 bytes	0.0295	0.1439	0.0254/86%	0.0245/17%	0.0035/12%	0.058/4%	4.1649
	32 bytes	0.0288	0.1428	0.0251/87%	0.0228/16%	0.0029/10%	0.0065/5%	4.1890
	64 bytes	0.0280	0.1400	0.0244/87%	0.0233/17%	0.0032/11%	0.0048/3%	4.2015

thus improving processor performance dramatically. On the other hand, SMT has a shortcoming that inter-thread cache interference, which is detrimental to overall performance, increases due to multithreading. However, the impact that the cache configuration has on inter-thread cache interference in the SMT architecture has not been fully analyzed.

This paper categorizes cache misses as first-reference misses, inter-thread conflict misses, and intra-thread conflict misses, and we measure the impact that the size, set associativity and block size of caches have on the cache interference and processor throughput of the SMT processors. The size of caches affects both inter-thread and intra-thread cache interference, and has a large impact on the performance of SMT processors. The set associativity of caches has a significant impact on inter-thread cache interference and processor throughput, but has a limited impact on intra-thread cache interference. The block size of caches has a relatively small impact on cache misses and performance.

Besides, this paper somewhat overstates the amount of inter-thread cache interference, since we have not applied compiler optimizations to reduce interference. We believe that compiler optimizations exist that reduce inter-thread cache interference in the SMT architecture, and this is an area of future research.

Acknowledgements

The work reported in this paper was supported by the Brain Korea 21 Project in 2004.

References

[1] P. Song, "Multithreading comes of age," *Microprocessor Report*, Vol. 11, No. 9 July 14, 1997.

[2] D. M. Tullsen, S. J. Eggers, and H. M. Levy, "Simultaneous multithreading: maximizing on-chip parallelism," *Proc. 22nd International Symposium on Computer Architecture*, pp. 392–403, Santa Margherita Ligure, Italy, June 1995.

[3] J. L. Lo, S. J. Eggers, J. S. Emer, H. M. Levy, R. L. Stamm, and D. M. Tullsen, "Converting thread-level parallelism to instruction-level parallelism via simultaneous multithreading," *ACM Transactions on Computer Systems*, Vol. 15, No. 3, pp. 322–354, Aug. 1997.

[4] R. Thekkath, and S. J. Eggers, "Impact of sharing-based thread placement on multithreaded architectures," *Proc. 21st International Symposium on Computer Architecture*, pp. 176–186, Chicago, Illinois, April 1994.

[5] J. L. Lo, L. A. Barroso, S. J. Eggers, K. Gharachorlo, H. M. Levy, and S. S. Parekh, "An analysis of database workload performance on simultaneous multithreading processors," *Proc. 25th International Symposium on Computer Architecture*, pp. 39–50, Barcelona, Spain, June 1998.

[6] B. I. Moon, *Study of an In-order SMT Architecture and Grouping Schemes*, Ph.D. Thesis, Yonsei University, Seoul, Korea, 2002.

[7] B. I. Moon, M. G. Kim, I. P. Hong, K. C. Kim, and Y. S. Lee, "Study of an in-order SMT architecture and grouping schemes," *International Journal of Control, Automation, and Systems*, Vol. 1, No. 3, 339–350, Sept. 2003.

[8] D. Jagger, and D. Seal, *ARM Architecture Reference Manual*, 2nd Edition, Addison-Wesley, Boston, Massachusetts, 2000.

[9] J. L. Henning, "SPEC CPU2000: measuring CPU performance in the New Millennium," IEEE Computer, Vol. 33, No. 7, pp. 28–35, July 2000.

[10] ARM Limited, *ARM Developer Suite*, Nov. 2001, http://www.arm.com/documentation/Software_Development_Tools/index.html.