Shared Cache Allocation Based on Fairness in a Chip Multiprocessor Architecture

Dongfang Wang(✉) and Jingmei Li

College of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China
{wangdongfang,lijingmei}@hrbeu.edu.cn

Abstract. Cache, as the hub between the multi-core processor and the memory, is closely related to the performance of the CMP system. And with the number of processor core increasing, the contention of multi-core for shared cache becomes more intense. Fairness is important to affect the performance of CMP systems, so a Shared Cache allocation method based on fairness is proposed. According to the way of borrow-return, the method assigns Shared Cache to multiple to realize the dynamic balance. The experimental results show that the method significantly improves the system fairness and the system throughput.

Keywords: Multi-core · Shared cache · Fairness · Allocation

1 Introduction

At present, the design of on-chip multi-core (CMP) is widely used in processor design, and the cache structure is also developed from the previous one level cache to multi-level cache design [1]. The paper studies the two level cache. CMP architecture mostly uses private L1 cache and shared L2 cache to improve resource utilization [2]. However, the contention of multi-core for shared cache becomes more intense. There are some problems. On the one hand, due to the interference between threads, one thread may replace another thread's "hot" data, which causes the thread data access failure [3]. On the other hand, because of shared Cache space competition, some that can quickly produce a large number of cache invalidation thread may replace the valid data of other threads, exclusive most or all of the cache space. Therefore, each core uses fair share cache as the research objective, and proposes a shared cache allocation method. The method assign the shared cache to more than one core. And the core which have the phenomenon of frequent data access failure can borrow cache form other cores, paying back after a while.

2 Shared Cache Allocation

Shared Cache allocation is defined as each core privately occupies part of Shared Cache by certain rules. Once a part of the cache space is assigned to a core, the core has control over its cache space. The Shared Cache allocation is divided into static and dynamic. Compared with dynamic allocation, static allocation distribute cache without considering the dynamic demand for cache in program runtime [4]. So dynamic

G. Sun and S. Liu (Eds.): ADHIP 2017, LNICST 219, pp. 501–504, 2018.
https://doi.org/10.1007/978-3-319-73317-3_58

allocation is better. In the process of shared Cache allocation, fairness affects the performance of the system [5]. If fairness is ignored, certain thread access requests may not be answered for long time, even starvation. Therefore, the paper proposes a shared cache allocation method based on fairness.

There are five 5 metrics defined by kim [6] that measure the fairness of CMP systems.

$$M_1^{ij} = \left|X_i - Y_j\right| \quad X_i = \frac{Miss\_shr_i}{Miss\_ded_i} \tag{1}$$

$$M_2^{ij} = \left|X_i - Y_j\right| \quad X_i = Miss\_shr_i \tag{2}$$

$$M_3^{ij} = \left|X_i - Y_j\right| \quad X_i = \frac{Missr\_shr_i}{Missr\_ded_i} \tag{3}$$

$$M_4^{ij} = \left|X_i - Y_j\right| \quad X_i = Missr\_shr_i \tag{4}$$

$$M_5^{ij} = \left|X_i - Y_j\right| \quad X_i = Missr\_shr_i - Missr\_ded_i \tag{5}$$

Miss_shr$_i$ denote the number of misses of thread i when it shares the cache with other threads. And Miss_dedi denote miss rate of thread I when it runs alone with dedicated cache. M1is used to balance the increase in the number of cache failures per thread. M2 is the number of cache failures that balance each thread. M3 is used to balance the percentage of thread cache failure rates increasing. M$_4$ balances cache failure rates for each thread. M$_5$ is used to balance the cache failure rate due to coordinated running of threads.

To illustrate the problem, N denote the number of date block of cache. M denote the number of core. P$_i$ denote the cache block assignment information table of core$_i$. Q$_i$ denote the cache block debit table of core$_i$. The allocation of shared cache is mainly divided into initialization, borrowing and repayment phases. The specific steps are as follows:

Initialization:

1. Initializes the table Q and table P. Share the shared cache blocks N equally to the M cores. Every core update its cache block assignment information table.

Lending step:

2. When the core$_i$ sends access request to the Cache, the address of the Cache block is obtained by mapping rules according to the address to be accessed, and judge whether the cache block address is hit in the table P$_i$. if hit, jump step 3.else jump step 7.
3. To determine whether data blocks frequently change in and out, if there is a frequent replacement of data blocks, jump step 4. Else jump step 5.
4. Determine whether the debit table core$_i$ of Q$_i$ is empty, if not empty, then take back the cache block from the debit table Q$_i$, and execute step 5. If the debit table Q$_i$ is empty, then borrow the core$_j$ which has relatively ample cache space cache block

and start the timing. Add these cache block address into P_i and Q_j. And delete these cache block information from P_j and jump step 6.

5. Execute the replacement policy of data, and update the assignment information table P_i of $core_i$.

Payment step:

6. When the timing reaches the threshold, $core_i$ returned cache block which borrowed from $core_j$ to the $core_j$. And delete the information of borrowed cache block from table P_i and Q_j. Then restore the borrowed cache block information to the table P_j.

3 Evaluation

To evaluate the benefit of the Shared Cache allocation based fairness. We choose a part of SPEC CPU 2006 benchmarks that are memory–intensive. The evaluation is using simulator M5. The CMP cores are set private L1 instruction and data, and shared L2 cache. Processor employs 4-core, Out-of-order, 1 GHz. The Protocol adopts a simple snooping cache coherence protocol. And L1 Cache is designed private 2-way 32 KB DC/32 KB IC block size = 64 B. L2 Cache is designed shared 8-way, block size = 64 B.

3.1 Simulation Result

Figure 1 shows the throughput (combined IPC) of fairness partitioning and no-fairness partitioning. From the average IPC contrast of each test program in the figure, we can see that the fairness partitioning is better than the no-fairness partitioning. Compared with no-fairness partitioning, IPC of the fairness partitioning improved by 9.7%.

Figure 2 shows system miss rate result which was LRU policy and LRU based on the fair shared cache partitiong. On the whole, the miss rate of system that employs fair shared cache partitioning is obviously decreasing. And the miss rate of LRU that

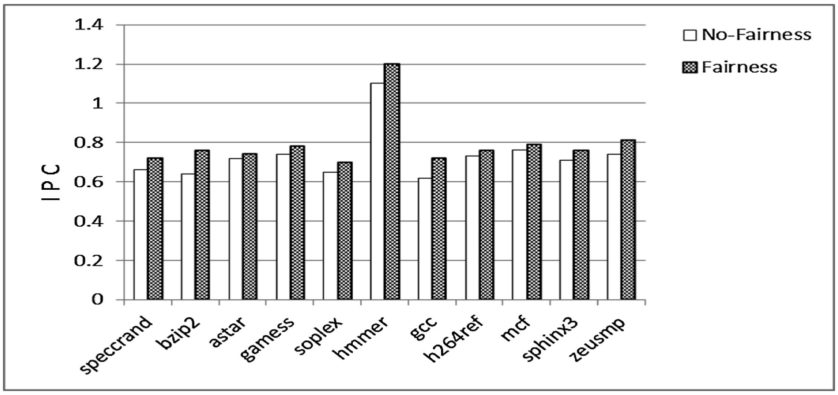

Fig. 1. System throughput comparison

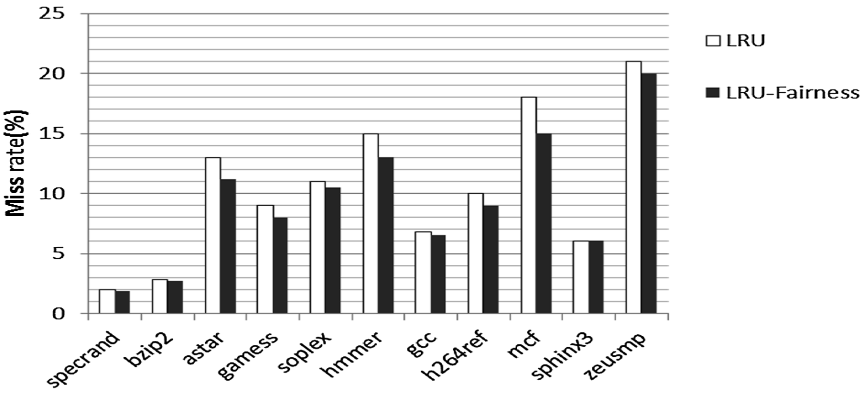

Fig. 2. System miss rate comparison

employs fair shared cache partitioning is decrease by 15% compared with the miss rate of LRU that does not employ shared cache partitioning.

4 Conclusions

The paper has shown the comparison of system throughput and miss rate between shared cache allocation based fairness and no-fair shared cache allocation. The method proposed by the paper uses the fairness as benchmark to realize the dynamic balance of cache allocation. The experimental results show that the IPC of system is greatly improved and the performance of the system is better. And with fair cache partitioning, the OS can assign Shared Cache reasonably to the cores that really need it.

References

1. Pan, A., Pai, V.S.: Runtime-driven shared last-level cache management for task-parallel programs. In: 2015 SC - International Conference for High Performance Computing, Networking, Storage and Analysis, no. 11. IEEE (2017)
2. Mars, J., Hundt, R., Vachharajani, N.A.: Cache contention management on a multicore processor based on the degree of contention exceeding a threshold (2016)
3. Das, S., Kapoor, H.K.: Towards a better cache utilization by selective data storage for CMP last level caches. In: International Conference on VLSI Design and 2016 International Conference on Embedded Systems, pp. 92–97 (2016)
4. Chang, J., Sohi, G.S.: Cooperative cache partitioning for chip multiprocessors. In: International Conference on Supercomputing, pp. 242–252. ACM (2007)
5. Brock, J., Ye, C., Ding, C., et al.: Optimal cache partition-sharing. In: International Conference on Parallel Processing, pp. 749–758. IEEE (2015)
6. Kim, S., Chandra, D., Yan, S.: Fair cache sharing and partitioning in a chip multiprocessor architecture. In: 2004 Proceedings of International Conference on Parallel Architecture and Compilation Techniques, PACT 2004, pp. 111–122. IEEE (2004)