Overview of the Typical Methods of Shared Cache Partitioning for Multi-core

Mingquan Zhang

School of Control and Computer Engineering, Engineering Research Center of Intelligent Computing for Complex Energy Systems, Ministry of Education, North China Electric Power University, Baoding 071003, China.

Abstract

Multi-core processor can greatly improve the system performance because of its parallelism. It has been widely used and gradually replaced the single-core processor. Different from single-core and single-task processing, tasks running on multi-core processors are mostly parallel processed, which will involve shared resource conflicts. Cache is an important measure to improve the speed of memory access. Except to the small capacity exclusive cache inside the CPU, large capacity cache generally exists in the form of multicores sharing. This paper analyzes and summarizes different kinds of typical cache partitioning methods, which can improve the overall performance or/and power or/and fairness of the system.

Keywords

Shared Cache; Partitioning; Multi-core.

1. Introduction

Multi-core processor on-chip was first proposed by Stanford University in the United States. Its idea is to integrate symmetric multicores in massively parallel processors into the same chip, and each core executes different processes in parallel. In the early days, it was only used on the server. Because of the backward manufacturing technology, the server was expensive and the application area was narrow, the multi-core processor was not popularized. Later, with the improvement of chip manufacturing technology and the decline of price, the multi-core processor was also used on the personal computer and developed rapidly.

In a multi-core environment extra misses are endured in the last level shared cache because each process is trying to utilize the whole cache space, resulting in high interference between their working sets, and thus a degraded performance. Therefore,scholars at home and abroad have proposed a variety of new technologies to optimize the cache allocation under the multi-core architecture, including the design of hardware and software. Finally, the performance, overall fairness and energy consumption of the system have been greatly improved. This paper will analyze and summarize the existing cache partitioning technology, and provide the basis for the development of new cache partitioning technology.

2. Typical methods of shared cache partitioning

2.1 Performance oriented cache partitioning.

With the development of multi-core processor architecture, the conflict caused by shared cache among parallel tasks becomes the bottleneck of performance improvement. In [1], they proposed utility-based cache partitioning (UCP), a low-overhead, runtime mechanism that partitions a shared cache between multiple applications depending on the reduction in cache misses that each application is likely to obtain for a given amount of cache resources. The proposed mechanism monitors each application at runtime using a cost-effective, hardware circuit that requires less than 2kB of storage. The information collected by the monitoring circuits is used by a partitioning algorithm to decide the amount of cache resources allocated to each application. Their evaluation, with 20 multiprogrammed workloads, shows that UCP improves performance of a dual-core system and an LRU-based cache

partitioning. Aaron Lindsay et al. [2] present a partitioning scheme called LWFG, which minimizes cache misses by partitioning tasks that share cache onto the same core and by evenly distributing the total working set size across cores. Their implementation reveals that LWFG improves execution efficiency and reduces mean maximum tardiness over past works.

Sukarn Agarwal et al. [3] reduce the miss penalty and improve the average memory access time by retaining the victims evicted from the hybrid cache in a smaller, fully associative SRAM structure called the victim cache. The victim cache is accessed on a miss in the primary hybrid cache. Hits in the victim cache require an exchange of the block between the main hybrid cache and the victim cache. In such cases, to effectively place the required block in the appropriate region of the main hybrid cache, they propose an access based block placement technique. Besides, to manage the run time load and the uneven evictions of the SRAM partition, they also present a dynamic region-based victim cache partitioning method to hold the victims dedicated to each region. Their experimental evaluation shows significant improvement in the performance and execution time along with a reduction in the overall miss rate. In [4] a cache-friendly algorithm is proposed, their algorithm is able to achieve near-ideal scaling in practice by avoiding the memory-bandwidth bottleneck. And the algorithm's performance is comparable to that of the previous excellent Blocked Strided Algorithm for parallel EREW sorting algorithms.

2.2 Low power oriented cache partitioning

Multi-core architectures, especially chip multi-processors (CMP), have been widely acknowledged as a successful design paradigm. Dynamic cache reconfiguration is a promising technique in reducing energy consumption of the cache subsystem for uniprocessor systems. In [5], they present an energy optimization technique which employs both dynamic reconfiguration of private caches and partitioning of the shared cache for multi-core systems with real-time tasks. Their static profiling based algorithm is designed to judiciously find beneficial cache configurations of private caches for each task as well as partition factors of the shared cache for each core so that the energy consumption is minimized while task deadline is satisfied. Experimental results using real benchmarks demonstrate that their approach can achieve substantial energy saving on average compared to systems employing only cache partitioning.

To reduce energy consumption in CMP, Fang et al. [6] propose a mechanism based on dynamical way-adaptable cache. The mechanism mainly consists of way reallocate module and dynamic power control module. Way reallocate module reassigns ways between cores based on thread’s working set on the execution of the program. Their mechanism implements low power consumption by dynamic power control module. Compared with previous traditional cache, their proposed scheme based on dynamical way-adaptable cache can increase the performance and reduce power consumption. In [7] a reuse locality aware cache partitioning scheme (ROCA) to reserve higher reuse locality blocks in the last level shared cache is proposed. ROCA uses a reuse locality reservation algorithm to maintain the cache fraction. Meanwhile, reuse locality guided block placement policy is proposed to further improve cache efficiency. Evaluation results show that their proposed technique improves energy efficiency as well as system performance with reduced hardware cost, compared to other similar methods.

In [8], for the shared cache energy in hard real-time multicores with IABA (interference-aware bus arbiter) bus, a bank-column partitioning optimization method is proposed. Their method ensures that hard real-time tasks can be completed before the deadline. And the experimental results show that their proposed optimization method can reduce the energy consumption of shared cache on the premise that the hard real-time tasks can be completed before the deadline.

2.3 Fairness oriented cache partitioning

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. In [9], 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 their method significantly improves the system fairness and the system throughput.

Shahira Abousamra, et al.[10] propose a cache partitioning scheme, which opposed to prior partitioning schemes in the literature, offers both high performance and fair allocation even with increasing number of simultaneously running processes and decreased cache associativity. Their proposed scheme takes advantage of the broad number of sets in the cache by partitioning it dynamically set-wise. The partitioning presented is adaptive to accommodate to the changing requirements of the mix of running processes without the need for any offline information and with an eye on fair allocation of cache resources. Performance is evaluated by experimentation on different cache sizes with different associativity and comparison with the Least Recently Used baseline cache and partitioning schemes. Their experimented results show that the performance and the fairness are both improved.

In [11] a partition-based cache management algorithm for a shared cache is proposed. The goal of their algorithm is to find an allocation such that all heterogeneous applications can achieve a specified fairness degree as least performance degradation as possible. To achieve this goal, they present an adaptive partition framework, which partitions the shared cache among competing applications and dynamically adjusts the partition size based on predicted utility on both fairness and performance. Their experimental results show that, compared with LRU, their algorithm achieves large improvement in fairness and slightly in performance.

2.4 Resource scheduling oriented cache partitioning

The current trend in mobile computer devices systems is to integrate more software applications into fewer cores to decrease the cost and increase efficiency. This means more applications share the same resources which in turn can cause congestion on resources such as caches. Shared resource congestion may cause problems for time critical applications due to unpredictable interference among applications. It is possible to reduce the effects of shared resource congestion using cache partitioning techniques, which assign dedicated cache lines to different applications. In [12], they propose a cache partition controller called LLC-PC that uses the Palloc page coloring framework to decrease the cache partition sizes for applications during runtime. LLC-PC creates cache partitioning directives for the Palloc tool by evaluating the performance gained from increasing the cache partition size. They show that LLC-PC is able to decrease the amount of cache size allocated to applications while maintaining their performance allowing more cache space to be allocated for other applications.

The dynamic scheme has proficiency in schedulability, flexibility, and energy-efficiency. In [13], Sheikh et al. make initial contributions to a dynamic cache-partition schedulability analysis for multicore partitioned scheduling of real-time fixed-priority sporadic tasks. They devise a sufficient schedulability test and then refine the upper-bound by proposing techniques to reduce the pessimism in the interference caused by cache-contention.

Ohchul Kwon et al. [14] present a cache partition allocation framework enabling flexible cache partitioning for multi-mode real-time systems. Their main objective is to guarantee timing predictability in the steady states and during mode changes. They evaluate the effectiveness of their approach for multiple embedded benchmarks with different ranges of cache size sensitivity. Their results show increased schedulability compared to static partitioning approaches.

3. Summary

For a multi-core system with shared cache, allocating and managing limited cache resource is a very important and challenging problem. With the rapid growth of the number of processor cores, the competition between cache access requests from different threads becomes more and more fierce, which leads to more significant impact on system performance, energy and system fairness. In addition to increasing the available shared cache, fair and efficient management and utilization of

shared cache is very important. In this paper, we summarize the typical shared cache partitioning methods designed for different purposes.

Acknowledgments

This paper was financially supported by “the Fundamental Research Funds for the Central Universities (2020MS122)”.

References

[1] MK Qureshi et al. Utility-Based Cache Partitioning: A Low-Overhead, High-Performance, Runtime Mechanism to Partition Shared Caches[C]//The 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'06),2006

[2] Aaron Lindsay et al. On Cache-Aware Task Partitioning for Multicore Embedded Real-Time Systems.[C]// IEEE Intl Conf on High Performance Computing and Communications; IEEE International Conference on embedded Software and Systems; International Symposium on Cyberspace Safety and Security. 2014. 677-684

[3] AgarwalSukarn, Kapoorhemangee K. Improving the Performance of Hybrid Caches Using Partitioned Victim Caching[J]. ACM Transactions on Embedded Computing Systems (TECS), 2020.

[4] William Kuszmaul et al. Cache-Efficient Parallel-Partition Algorithms using Exclusive-Read-and-Write Memory[C]//SPAA '20: 32nd ACM Symposium on Parallelism in Algorithms and Architectures. July 2020

[5] Wang W, Mishra P, Ranka S. Dynamic cache reconfiguration and partitioning for energy optimization in real-time multi-core systems[C]// Proceedings of the 48th Design Automation Conference, DAC 2011, San Diego, California, USA, June 5-10, 2011. ACM, 2011.

[6] FANG Juan et al. Research on path adaptive partitioning algorithm for low power shared cache [J]. Computer science 2014 No. 7 36-39,73.

[7] Fanfan Shen, et al. Reuse locality aware cache partitioning for last-level cache[J]. Computers & Electrical Engineering, 2019, 74:319-330.

[8] Zhang Jizan, yuan Yajuan. A multi-core shared cache energy optimization method based on bank column cache partition [J]. Journal of Xinyang Normal University: Natural Science Edition, 2017 (4)

[9] Wang D, Li J. Shared Cache Allocation Based on Fairness in a Chip Multiprocessor Architecture [J]. 2017.

[10] Shahira Abousamra, et al. Fair and adaptive online set-based cache partitioning[C]. The 2011 International Conference on Computer Engineering & Systems.2011

[11] Li Y, Feng D, Shi Z. Heterogeneous-aware cache partitioning: Improving the fairness of shared storage cache[J]. Parallel Computing, 2014, 40(10):710-721.

[12] Danielsson J, Jagemar M, Behnam M, et al. Run-Time Cache-Partition Controller for Multi-Core Systems[C]// IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society. IEEE, 2019.

[13] Sheikh, Saad & Pasha, Muhammad. A Dynamic Cache-Partition Schedulability Analysis for Partitioned Scheduling on Multicore Real-Time Systems[J]. IEEE Letters of the Computer Society. PP. 1.2020.

[14] Kwon O, et al. Flexible Cache Partitioning for Multi-Mode Real-Time Systems[C]// Design, Automation and Test in Europe Conference (DATE 2021). 2021.