Power-aware Event-driven Cache Partitioning for high performance chip multiprocessor

Myeongjin Kim, Byunghoon Lee, Minsik Oh, and Eui-Young Chung
Department of Electrical and Electronic Engineering
Yonsei University
Seoul, Korea
kmjjang86@gmail.com, {bh2, stomlions}@dtl.yonsei.ac.kr, eychung@yonsei.ac.kr

Abstract*—For the high performance of chip multiprocessors (CMP), maximizing the utilization of shared resources has been one of challenging works in recent years. Shared cache with cache partitioning is one of great solution for CMP which allocates fitted cache size to each core in CMP. This paper introduces event-driven cache partitioning which dynamically reconfigures way-aligned cache partitioning and also considers shared accesses between multiple cores. Moreover, it reduces the static power by transferring redundant ways into drowsy or power-down mode. We verify that our proposed method sufficiently guarantees high hit rate and obtains more power saving of 34.54% and 23.74% than baseline cache and state-of-the-art method, although it needs less reconfiguration overhead.*

Keywords—cache partitioning; hit rate; power saving; reconfiguration overhead

I. INTRODUCTION

Most of recent computing devices have employed chip multiprocessor (CMP) for handling the multiple applications. To successfully drive CMP architecture, it is significant to manage shared cache well not to compete with each internal core in CMP.

Cache partitioning, one of widely used scheme for shared cache, arranges a suitable cache size with reference to application's characteristic and allocates to corresponding core. The conventional cache partitioning adopts interval-based method which makes a decision for cache reconfiguration every fixed interval. However, it becomes a big burden because of too redundant cache partitioning.

In this paper, we propose event-driven cache partitioning that contributes to both system performance and power saving with little hardware and decision overheads.

II. RELATED WORKS

Dynamic cache partitioning has been widely proposed to improve the drawback of static cache partitioning. The author in [1] introduced monitoring unit of cache utilization for cache partitioning. Required hardware burden, however, should be larger along with large cache size. The author in [2] proposed power efficient cache partitioning using power-down mode to unused cache ways. When the way enters into power-down mode, its cell information is flushed and this may cause an unexpected cache miss in future access.

Our proposed method remarkably minimizes cache reconfiguration overhead using event-driven approach and introduces two kinds of low power mode, drowsy and power-down mode, to prevent unexpected performance degradation. In addition, we take account of shared way to compensate cache conflict problem that aforementioned methods [1-2] do not contain.

III. PROPOSED METHOD

Dynamic cache partitioning requests cache reconfiguration in processing time unlike static cache partitioning. To make an accurate decision for cache reconfiguration, we propose event-driven cache partitioning. There exist following three events that trigger cache reconfiguration.

- Miss event ($Event_M$) – When a variation of miss rate in any partitioned cache area is higher than a predefined threshold value.
- Conflict event ($Event_C$) – When a cache thrashing level is higher than a predefined threshold value.
- Time-out event ($Event_T$) – When an access frequency of certain way is lower than a predefined threshold value.

Based on three events, cache reconfiguration is done by way transition between four kinds of following way pools.

- Private way pool ($Pool_P$) – The set of ways only core *i* can access. There are *n* private way pools if number of cores in CMP is *n*.
- Shared way pool ($Pool_S$) – The set of ways all of cores in CMP can access.
- Drowsy way pool ($Pool_D$) – The set of ways in drowsy mode. The ways in $Pool_D$ preserve the current cell state.
- Power-down way pool ($Pool_{PD}$) – The set of ways in power-down mode. The ways in $Pool_{PD}$ destroy the current cell state.

Fig. 1 shows the way transition triggered by three events. First, $Pool_P$ acquires an additional way from ①$Pool_D$ ②$Pool_{PD}$ ③ other $Pool_P$ when $Event_M$ is triggered. To prevent

performance degradation, $Pool_P$ searches the idle way in drowsy mode first which maintains the previous cell state. The hit rate of way from drowsy mode is also higher than that of power-down mode. If there is no idle way in $Pool_D$ and $Pool_{PD}$, $Pool_P$ takes the way from other $Pool_P$ which has sufficiently high hit rate.

Second, $Pool_S$ acquires an additional way from ①$Pool_D$ ② $Pool_{PD}$ ③other $Pool_P$ when $Event_C$ is triggered. The reason for this acquisition ordering is same with $Pool_P$.

Lastly, the redundant ways in $Pool_P$ and $Pool_S$ are transferred to ①$Pool_D$ ②$Pool_{PD}$ when $Event_T$ is triggered. If the way triggers $Event_T$ at first time, it moves into $Pool_D$ first to prepare for consecutive $Event_M$ and $Event_C$. If $Event_T$ is triggered once again, it moves into $Pool_{PD}$ for aggressive power saving.

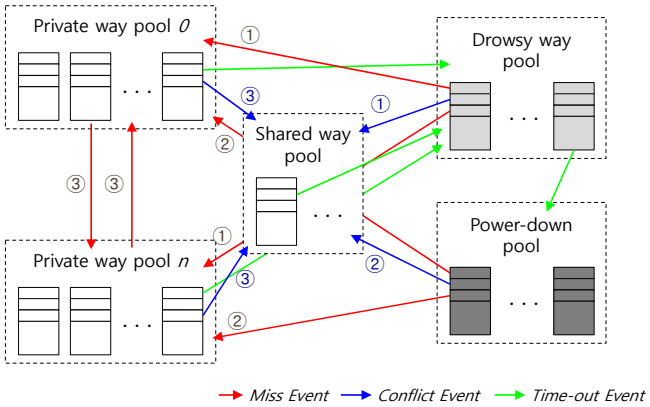

Fig. 1. Way transition between way pools

IV. EXPERIMENTS

We evaluated our cache partitioning method on the stand-alone cache simulator we designed. Cacti [3] is invoked in early stage of our simulator to use the physical characteristic of cache. The cache traces are extracted from full system simulator, GEM5 [4], by running various combinations of application in Parsec benchmark suite [5].

The traces are sorted into multi-programmed (MP) and multi-threaded (MT) group. MP means multiple applications are concurrently executed on multiple cores and MT means one application is partially executed by multiple threads, i.e. one thread per core in our scenarios. To verify with other methods, we implemented baseline cache (BC), static cache partitioning (SCP), UMON-based cache partitioning (UCP) [1], Cooperative cache partitioning (CCP) [2].

Fig. 2 shows the hit rate of last-level cache (LLC). In MP group, SCP is worse than BC since it cannot reflect the variation of combined applications. By comparison, UCP, CCP and proposed method provide higher hit rate due to its dynamic approach. However, in MT group, only proposed method shows better result than BC. This is owing to the consideration of shared way pool.

The average power consumption of LLC is depicted in *Fig. 3*. Both of SCP and UCP do not support low power mode and consume same power as much as BC does. CCP and proposed method, however, turn off redundant ways to reduce static power. Proposed method obtains the power saving of 34.54% compare to BC. Furthermore, proposed method shows more reduction of 23.74% than CCP, since the event-driven method notifies an accurate timing of transition to low power mode.

Proposed method needs a hardware overhead of 88 bytes to event monitoring, whereas 8,720 bytes in UCP. (4 cores in CMP and 16 ways in LLC in our scenarios, and more details in *TABLE I*) Its decision overhead is also reduced to 55% compared to CCP by pruning redundant cache reconfigurations.

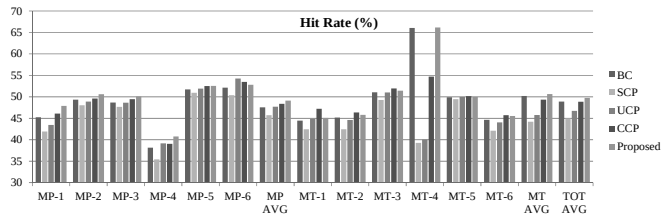

Fig. 2. Hit rate of last level cache

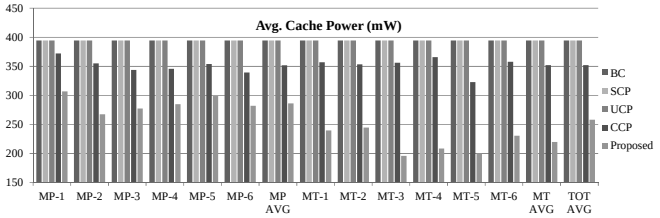

Fig. 3. Average power consumption of last level cache

TABLE I. HARDWARE OVERHEAD OF PROPOSED METHOD

Overhead	**Proposed method**	
Monitoring overhead	Access counter (16 ways * 4 bytes)	64 bytes
	Miss counter (4 cores * 4 bytes)	16 bytes
Reconfiguration overhead	Access Permission Register (4 cores * 16 ways)	8 bytes
Total overhead	-	**88 bytes**

V. CONCLUSION

Event-driven cache partitioning we have proposed makes a great contribution for high performance CMPs with little cache reconfiguration overhead. Experimental results demonstrate that cache hit rate of proposed method is well guaranteed, especially when shared accesses are frequently occurred. It also gives more power savings than other cache partitioning methods. We think that cache partitioning will serve more benefit if there are much more shared cache levels between L1 cache and LLC.

ACKNOWLEDGMENT

This work supported in part by Basic Science Research Program through the National Research Foundation of Korea

(NRF) funded by the Ministry of Education (2013R1A1A2011208) and by LG Electronics.

REFERENCES

[1] M. K. Qureshi, and Y. N. Patt, "Utility-based cache partitioning: A low-overhead, high-performance, runtime mechanism to partition shared caches," in Proc. Int. Symp. on Microarchitecture, 2006.

[2] K. T. Sundararajan, V. Porpodas, T. M. Jones, N. P. Topham, and B. Franke, "Cooperative partitioning: Energy-efficient cache partitioning for high-performance CMPs," in Proc. Int. Symp. on High Performance Computer Architecture, 2012.

[3] S. Thoziyoor, N. Muralimanohar, J. H. Ahn, and N. P. Jouppi, "CACTI 5.1," HP Laboratories, April 2008.

[4] N. Binkert, et al., "The gem5 simulator," ACM SIGARCH Computer Architecture News, Vol. 39, No. 2, pp. 1-7, May 2011.

[5] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC benchmark suite: Characterization and architectural implications," in Proc. Int. Conf. on Parallel architectures and compilation techniques, 2008.