A Research on Practicality of Methods of Placing Cache Blocks

Takayuki Hiroyama (1010056)

School of Information Science,
Japan Advanced Institute of Science and Technology

February 6, 2013

Keywords: Hierarchical Caches, Multicore, Multiprocessor.

1 Background

There are two types of block-placing methods for hierarchical caches; Inclusion Property and Exclusion Property. In the Inclusion Property, the lower-level caches include copies of blocks that exist in the upper-level caches. For example, in multi-/many-core processors where each core has its own L1 cache and all cores share an L2 cache, the L2 cache always include copies of all the blocks in all the L1 caches. Therefore, this method puts pressure on the L2 cache capacity. This method has a merit that references for cache coherence management can be limited to the L2 cache, which leads to small overheads. On the other hand, in Exclusion Property, the upper-level and lower level caches have different blocks. Therefore, the L2 capacity is not wasted by the blocks in the L1. However, not only L2 but5 L1 caches must be referenced for cache coherence, which leads to large overheads.

As a research focusing on this tradeoff, Younsuk proposed a method of efficiently placing blocks in hierarchical cache systems [1], where categorization of memory references is proposed. The categorization focuses on locality of references for blocks to be placed in cache memories. However, the proposed method could not exhibit potential ability of such categorizations.

2 Related Work

Based on locality of blocks, reference patters are analyzed and blocks are located in the appropriate cache hierarchy. The analysis of reference patterns is performed by using informations about access frequency and access intervals of blocks. Blocks that are frequently accesses and have short access intervals are located in L1 caches. Blocks that are accessed over long time are located in L2 caches. This strategy can avoid wasting cache capacity by not placing blocks in inappropriate cache hierarchy. In addition, blocks including shared data are always located in the lowest-level cache, which can alleviate overheads of cache coherence management, since the scope is limited to the lowest-level cache.

3 Methods of Placing Blocks

In this thesis, a method of improving cache performance is proposed. In the proposed method, whether blocks are shared between processors or not is decided and whether it is worth placing blocks in each cache hierarchy or not is decided. The sharing information can obtained by analyzing memory access traces. When a block is found to be shared between processors, the block is located in the lowest-level cache. From this policy, overheads of references for coherence management can be reduced. Whether a block is located or not in each hierarchy is decided by comparing miss rates for each hierarchy. As a result, blocks with high miss rates are not located in the cache hierarchy, which contributes to performance improvements.

4 Evaluation Environment

As the environment for evaluating the proposed method, the author made several programs; a program that obtains memory access traces from benchmark programs, a program that performs categorization for the obtained memory access traces, a simulator for the proposed method that inputs the category information and memory access traces, and a simulator for the existing Inclusion Property and Exclusion Property. In the evaluation, fft, radix, lu, and cholesky are used as benchmark programs. The evalua-

tion is based on the number of clock cycles taken in the simulations of the proposed method, Inclusion Property, and Exclusion Property.

5 Conclusion

Inclusion Property exhibits a waste of cache capacity, while Exclusion Property leads to large overheads of referencing all hierarchies for coherence. In order to solve these problems, the former research [1] provided an idea of the categorization for cache references. The objective of this thesis is to make clear the potential ability of the categorization.

In this research, the author made the simulator for evaluating the proposed method and evaluated the method by using benchmark programs and their memory access traces.

References

[1] HUH Younsuk, Efficient Block Distribution Method for the Hierarchical Cache Systems, Master Thesis, JAIST, 2011.

[2] S.C.Woo, M.Ohara, E.Torrie, J.P.Singh, and A.Gupta, SPLASH-2 Programs: Characterization and Methodological Considerations Proc. of ISCA pp.24–36, 1995.

[3] Intel, PIN, http://www.pintool.org