Efficient Block Distribution method for the Hierarchical Cache Systems

Huh Younsuk (0910055)

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

February 8, 2011

Keywords: Hierarchical cache, Block distribution, Multicore processor.

1 Background

Many modern cache designs use the Inclusion Property to manage their hierarchical cache. Because the lower level cache always holds copies of the upper layer cache, Inclusion property managed caches are very efficient in reducing the cache coherence complexity[1], but has a drawback of using extra cache capacity resources due to the redundant copies[2][3]. This was not a problem in the case of a single core processor due to the fact that the capacity of the lower level cache was very large compared to the capacity of the upper level cache. But in the case of a multi-core processor that each has an independent upper level cache, the total capacity of the upper level cache cannot be negligible in comparsion to the lower level cache. On the other hand, cache designs using the Exclusion Property makes the best use of the lower cache capacity by eliminating redundant copies on the lower level cache[2][3], but has a drawback of increased overhead in coherence management due to the complexity[1].

In this research I propose a method of block distribution for hierarchical caches to achieve efficient use of cache resources and reduced overhead by simple coherence management. I have implemented a simulator capable of simulating the proposed method and the conventional method. Using the simulator, I compared the results of the execution time and cache misses to evaluate the efficiency of the proposed method.

2 Block distribution using locality information

In this research, I propose a method of distributing data to the cache levels according to the data access patterns. Blocks that consist of data that have frequent and long term accesses are likely to be referenced soon after they have been evicted. Therefore, these blocks are distributed both to the upper level cache and the lower level cache. Blocks that consist of data that have infrequent accesses but have long term accesses are distributed to the lower level cache, and are not distributed to the uper level cache to avoid evicting blocks consisting of data with frequent accesses in the upper level cache. To deal with the spatial locality of these blocks, theses blocks are also distributed in the buffer. Blocks that consist of data that have frequent accesses but do have long term accesses are distributed to the upper level cache, and are not distributed to the lower level cache to avoid evicting blocks consisting of data with long term accesses in the lower level cache. Blocks that consist of data that are accessed only a few times are not distributed to the caches to avoid evicting blocks consisting of data with higher access rates. To deal with the spatial locality of these blocks, theses blocks are distributed to the buffer. Blocks containing data shared among processors are always distributed to the lower level cache to simplify the coherency management. Distribution to the upper level cache depends on the previously described access patterns. The proposed method uses a buffer memory. The buffer is accessed in parallel with the upper level cache, and deals with blocks consisting of data only with spatial locality. Therefore, a small buffer size consisting of only a few blocks is sufficient for this matter. We assume the size of the buffer able to have equivalent or faster access time in comparsion to the upper level cache.

3 Simulation

To evaluate the efficiency of the proposed method, I have implemented a multicore(2core) processor simulator capable of simulating the proposed method. The implemented simulator is based on a SPARC architecture version 8 instruction set[4]. The cache memory of the simulator is a hierarchy cache consisting of separated instruction / data L1 caches and a shared L2 cache with an additional data buffer. The SPLASH-2[5] kernel programs were used for the evaluation. Simulation of the benchmark programs was performed by executing two programs simultaneously, and by executing a single parallel program. The performance is illustrated by comparing the execution time and cache misses of the proposed method and conventional method.

4 Conclusion

In this research I proposed a method of efficient block distribution to reduce loss in capacity by redundant copies in Inclusion property caches, and solve complex coherence management in Exclusion property caches. Also, I have illustrated the efficiency of the proposed method by implementing a simulator of the proposed method and executing benchmark programs on the simulator.

References

[1] J.-L. Baer and W.-H. Wang. On the inclusion properties for multi-level cache hierarchies. *SIGARCH Comput. Archit. News*, 16:73–80, May 1988.

[2] Ying Zheng, B. T. Davis, and M. Jordan. Performance evaluation of exclusive cache hierarchies. In *Proceedings of the 2004 IEEE International Symposium on Performance Analysis of Systems and Software*, pages 89–96, Washington, DC, USA, 2004. IEEE Computer Society.

[3] Manoj Franklin Mohamed M. Zahran, Kursad Albayraktaroglu. Non-inclusion property in multi-level caches revisited, June 2007.

[4] *The SPARC Architecture Manual*, 1991,1992.

[5] Stanford parallel applications for shared memory (splash-2).