

A Study of the Effect of Prefetching in Shared-Memory Resource Contention

Tanima Dey Wei Wang Jack Davidson Mary Lou Soffa
University of Virginia

Managing contention for shared resources in chip multiprocessors has become very challenging as the number of cores and execution contexts scale up. Contention for the memory hierarchy resources, especially the shared caches, can severely degrade an application's performance and system throughput [2]. One of the important resources related to caching is hardware prefetcher whose effect on the shared-memory resource contention has not been fully explored. Hardware prefetchers can have potential impact on an application's performance depending on its data access pattern. If the application has sequential or strided data access pattern, the prefetchers can bring data in the caches in advance, reducing cache misses and improving performance. On the other hand, if the application's data access pattern does not follow any pattern, the prefetched data can pollute the cache, resulting in more cache misses and worse performance. In this research, we study and measure the prefetching usefulness on multi-threaded applications' performances while the applications contend for different shared-memory resources. Such a study provides insights in understanding applications' behaviors and helps to further improve an application's performance by dynamically adjusting and utilizing prefetching along with the contention mitigating techniques. We use the multi-threaded PARSEC benchmarks to measure the prefetching effect on several resources in the memory hierarchy, including L1-cache, L2-cache and Front Side Bus (FSB).

To determine the effect of prefetching on shared-resource contention in the memory hierarchy, we need a methodology for comparing the application's performance when the prefetcher is enabled and disabled, at the same time when there is contention for the targeted resource. In general, to measure contention for a particular shared resource, applications are run in two *resource* configurations. In the *baseline* configuration, application threads are mapped onto cores such that the threads do not share the targeted resource and run using dedicated resources. In the *contention* configuration, the application threads are mapped onto the cores such that the threads execute sharing the targeted resource. Because a multi-threaded application has multiple threads, there is possibility of contention when these threads share a resource.

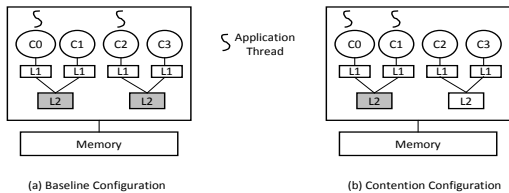


Figure 1: Configurations for measuring L2-cache contention

For example, to determine the effect of L2-cache contention on performance, we run each PARSEC benchmark in two configurations shown in Figure 1. As we measure L2-cache contention, we avoid L1-cache contention by allowing one thread to access one L1-cache and keep the bus contention unchanged by choosing a single-FSB platform. We calculate average performance difference between these configurations to measure L2-cache contention for each benchmark. We repeat these configurations once with the prefetchers turned on and once turned off. The performance difference in these two *prefetching* configurations deter-

mines prefetcher's role played in L2-cache contention. Using similar approaches, we can measure prefetching effects on L1-cache and FSB contention, with and without a co-running application.

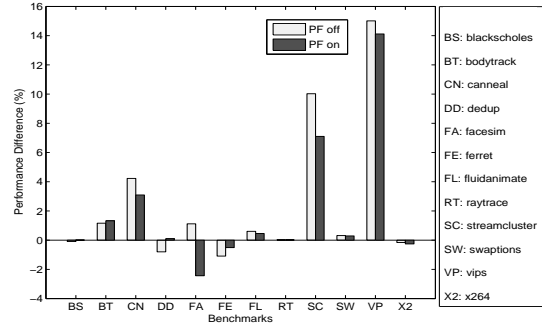


Figure 2: Experimental results of measuring prefetching effect on L2-cache contention

We performed several preliminary experiments following the above methodology (Figure 1) for L2-cache contention on Intel Q950 platform. To measure the performance, we utilized the hardware performance counter, UNHALTED_CORE_CYCLE. We used twelve benchmarks from the PARSEC2.1 benchmark suite [1], each with two threads and native input set. We collected the performance difference for five iterations in each *prefetching* configuration. The experimental results are given in Figure 2 where we observe that *CN*, *FL*, *SC*, *SW*, *VP* and *X2* show better performance for L2-cache contention when the prefetching is off. *FA*'s performance degrades by more than 4% when the prefetching is on. For these applications, the prefetchers do not find any data access pattern and pollute the L2-caches by the prefetched data. For *BT* and *FE*, the prefetchers cause very small performance improvement. *RT* is neither affected by L2-cache contention nor prefetching. Using a similar approach, we also measured prefetching effect on L2-cache contention for *SC* with several co-runners. *SC*'s performance degrades by 9% when the prefetching is on and runs with *VP*. It means the prefetched data of *VP*, *SC* or both pollute L2-cache for *SC*. In contrast, its performance improves by 2% when the prefetching is on and runs with *SW* which means *SW*'s prefetched data have no impact on *SC*'s performance and the performance improves because of *SC*'s own prefetched data. In summary, the prefetching effect varies across different applications and co-runners.

We plan to conduct more experiments to determine the prefetching effect on application's performance while they contend for the other shared resources (L1-cache, FSB) in the memory hierarchy, both with and without co-runners. The poster will include the new results and detailed analysis of the findings.

References

- [1] C. Bienia, S. Kumar, J. P. Singh, and K. Li. The PARSEC benchmark suite: Characterization and architectural implications. In *Int'l Conf. on Parallel Architectures and Compilation Techniques (PACT)*, 2008.
- [2] S. Zhuravlev, S. Blagodurov, and A. Fedorova. Addressing shared resource contention in multicore processors via scheduling. In *Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2010.