Reducing Memory Bandwidth for Chip-Multiprocessors using Cache Injection

Edgar A. León leon@cs.unm.edu
Arthur B. Maccabe maccabe@cs.unm.edu
Computer Science Department
University of New Mexico

Current and future high-performance systems will be constructed using multi-core chips. These systems impose higher demands to the memory system. Lack of adequate memory bandwidth will limit application performance. To reduce memory bandwidth we propose to use *cache injection of incoming network messages*. The objective of this work is to demonstrate benefits of cache injection and provide a basis for studying injection policies.

Current architectures place incoming network data into main memory and it is up to the processor to fetch data into a cache. Fetching data into a cache is usually done by prefetching which anticipates accesses to blocks of memory. Cache injection provides an alternative approach by placing incoming network data directly into a processor's cache. This technique reduces memory bandwidth by eliminating fetching data from main memory.

In this work, we quantify the difference between cache injection and prefetching. Through simulation we show: (1) cache injection significantly reduces memory bandwidth; and (2) cache injection outperforms prefetching on memory bandwidth. In addition, cache injection reduces execution time comparatively to prefetching.

Cache injection is not a new idea and has been used in previous work [2, 3] to improve uniprocessor application performance in commodity protocols (TCP/IP). Our study extends previous work by quantifying the difference between cache injection and prefetching in a zero-copy, OS-bypass environment which is typical for high-performance applications.

We evaluated cache injection and prefetching using IBM's PowerPC full-system simulator, Mambo [1], running the K42 research OS. We used a Power5-like architectural configuration with a cache injection implementation to the L3 cache.

Our evaluation consists of measuring memory bandwidth and execution time of an application in three configurations: (1) base case with no optimizations; (2) prefetching; and (3) cache injection. The application used in this evaluation is a user-level program that receives data from the network and reads every word of the incoming message sequentially. To evaluate cache injection in a suitable environment for high-performance applications, we implemented a zero-copy, OS-bypass messaging system based on UDP semantics [4].

In our first experiment, we measure the memory bandwidth used by the application in terms of the number of memory reads requested to the memory controller. The base case and prefetching perform equally as prefetching has to fetch incoming network data from memory. Prefetching anticipates data accesses correctly due to the sequential access pattern used by the application. Cache injection reduces the number of memory reads by 96% as all application accesses to incoming network data hit the L3 cache.

Our second experiment measures the execution time of the application in processor cycles. Both cache injection and prefetching outperform the base case as they both reduce the number of cache misses. Prefetching reduces execution time by 37% while cache injection by 30%. Prefetching performs better because it fetches blocks to the L2 and L1 caches, while our cache injection implementation targets the L3 cache. We expect that injections to the L2 will perform as good as prefetching or better.

Since cache injection reduces memory bandwidth for incoming network data, its benefits are directly proportional to the ratio of incoming network data and local data used by the application. In addition, cache injection benefits increase as the pressure on the memory bus increases (several processors).

Cache injection also presents some challenges. In our evaluation, the application uses the data shortly after it is injected into the cache. If the application does not use the data promptly, cache injection may create cache pollution taking the application's working set out of the cache. Thus, the performance benefits of this technique rely on a *good injection policy*. This policy is dependent upon the usage pattern of an application.

In conclusion, cache injection (1) alleviates memory bandwidth in a zero-copy, OS-bypass environment typical for high-performance applications, and (2) outperforms prefetching on memory bandwidth. In addition, cache injection has a positive effect on execution time by reducing execution time comparatively to prefetching. This work demonstrates benefits of cache injection on memory bandwidth and provides a basis for studying injection policies in a high-performance computing environment. Exploration of these policies remains as future work. We plan on using information from the OS and the compiler to make such decisions.

References

[1] Patrick Bohrer et al. Mambo – a full system simulator for the PowerPC architecture. *ACM SIGMETRICS Performance Evaluation Review*, 31(4):8–12, March 2004.

[2] Patrick Bohrer et al. Method and apparatus for accelerating Input/Output processing using cache injections, March 2004. US Patent No. US 6,711,650 B1.

[3] Ram Huggahalli et al. Direct cache access for high bandwidth network I/O. In *32nd Annual International Symposium on Computer Architecture (ISCA 2005)*, pages 50–59, Madison, WI, June 2005.

[4] Edgar A. León and Michal Ostrowski. An infrastructure for the development of kernel network services. In *20th ACM Symposium on Operating Systems Principles (SOSP'05). Poster Session*, Brighton, United Kingdom, October 2005. ACM SIGOPS.