Cache Injection for High-Performance Message-Passing Applications

Edgar A. León
Computer Science Department
University of New Mexico

Processor speeds increase at a faster rate than memory controllers. The effect of this growing disparity is known as the memory wall problem. The greater the disparity the more cycles wasted accessing data from memory. In applications where most of the data comes from the network, every access to memory is satisfied by the longest path in the memory hierarchy since I/O data is placed into main memory. These delays may yield ultimately to degrade the performance of the application. This problem presents an opportunity for improvement by moving I/O data closer to the processor in the memory hierarchy. In this paper, we present a framework to analyze this problem, and propose cache injection as a partial solution to the memory wall problem for message-passing applications. Cache injection allows network interface controllers to move data directly into a processor's cache decreasing latency, and possibly increasing performance by up to 30%.

Scientific applications demand tremendous amounts of computational capabilities. In the last few years, the computational power provided by clusters of workstations and SMPs has become popular as a cost-effective alternative to supercomputers. Nodes in these systems are interconnected using high-speed networks via "smart" Network Interface Controllers (NIC). These controllers allow the overlap of computation and communication by processing communication tasks on the NIC and computational tasks on the host processor(s).

Parallel applications running on these clusters suffer from a variety of performance and scalability problems: host processor overhead due to communication processing [7], data placement overhead, overhead due to external interrupts, cost of splitting OS functionality between host and NIC, etc. These problems are caused by the poor integration of the NIC with the Operating System (OS) and applications. The interactions between these entities are complex and are key to an application's performance.

This paper focuses on providing a framework where we can study how data placement of network data in the memory hierarchy affects application performance. Processor speeds increase at a much faster rate than memory. As these speeds increase, memory latency increases dramatically affecting application performance. Although processors may be fast, data cannot be accessed as fast as it may be produced by the network. In fact, the faster the processor, the more it has to wait to fetch data from main memory in terms of processor cycles.

To address this problem (and partially address the memory wall problem), we employ an architectural optimization called *cache injection* [3]. This optimization places network data closer to the processor in the memory hierarchy and thus reducing memory latency. In particular, it injects data from the network to a processor's cache directly. Cache injection has been studied recently to evaluate its impact on commodity environments using TCP/IP [4]. The intent of this study is to provide a framework where we can analyze its impact in a high-performance computing environment.

Our architectural assumption consists of a multiprocessor multi-core system (CMP - Chip-Level Multiprocessor). Multiple processor cores share a common second-level cache. Every core has a level-one cache and may support one or more logical threads. Second-level caches reside on the north side of the memory bus. Level-three caches and memory controllers reside on the south side. I/O devices (NIC and disk) are attached to the I/O bus and communicate with the memory through a bridge.

Cache injection is an architectural feature that allows I/O devices such as the NIC to initiate bus transactions directed to a target device. Without cache injection, this target device is the main memory controller. With cache injection, the transaction can be directed to a particular cache in the system. The target devices then snoop the bus to check whether they can match the target. If so, the target consumes the data, otherwise the memory controller does.

To investigate the effect of cache injection on a high-performance networking environment, we use: (1) an architecture that provides cache injection capabilities in hardware; and (2) a high-performance networking environment.

The former is provided by the IBM PowerPC full-system simulator, *Mambo*, and a networking module that allows the creation of NICs that can run arbitrary functionality [5]. This module can be *plugged* in to Mambo dynamically. Simulated NICs can be written in *C* and are accessed by the host through memory mapped registers. Using this mechanism, a user process can interact with the NIC directly, allowing *OS and Hypervisor bypass*. Access to these registers is controlled by the OS and/or Hypervisor [2].

To create a high-performance networking environment, we developed a simple high-performance messaging system called Fast UDP [6]. Fast UDP, a high-performance implementation of UDP, uses OS bypass to avoid the involvement of the OS in every message reception. Notification of message arrivals is implemented using a user queue that the application polls. This system has been implemented using *K42* [1], a high-performance research OS developed by IBM.

This infrastructure allow us not only to analyze cache injection but in general to analyze the trade-offs of moving OS functionality between the host and the NIC in a multiprocessor multi-core system. This is possible due to: (1) the ability of running OS services on the NIC; (2) the ability to define and simulate multiprocessor multi-core multi-threaded architectures in Mambo; and (3) the dynamic customization of system services in K42.

To characterize the potential benefits of cache injection in a

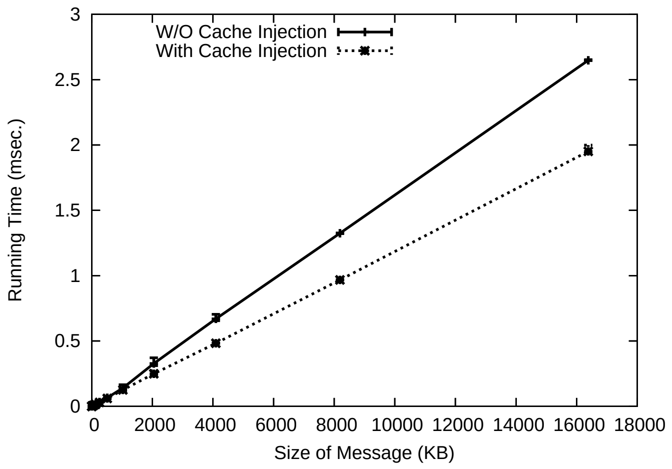

FIG. 1: Optimal Performance Improvement of Cache Injection Mechanism

high-performance networking environment, we created a simple user application in K42 and measured its runtime with and without cache injection. Cache injection is performed in a uniprocessor system to the L2 cache.

The application uses the Fast UDP communication library and consists of two phases: a communication phase, and a computation phase. During the communication phase, the process blocks waiting for network data to become available. In the computation phase, data is processed locally. By measuring the running time we quantify how much faster or slower the computation takes by having the data already in the cache. Figure 1 shows the initial results of this research. From this figure we see that as message sizes increase, application performance increases by as much as 30%. Furthermore, not shown in the figure, we see the expected performance decrease when injecting too much data in the cache, that may be taking out the working set of the application.

The contributions of this work are: (1) to provide an infrastructure to analyze cache injection for high-performance applications, and in general an infrastructure to better understand the interactions between the Operating System, Applications and smart NICs. This infrastructure allow us to identify, measure and analyze the bottlenecks in the data path from the network all the way to the application; and (2) propose cache injection as a solution to partially address the memory wall problem in a high-performance environment.

[1] Jonathan Appavoo, Marc Auslander, Maria Burtico, Dilma Da Silva, Orran Krieger, Mark Mergen, Michal Ostrowski, Bryan Rosenburg, Robert W. Wisniewski, and Jimi Xenidis. K42: an open-source linux-compatible scalable operating system kernel. *IBM Systems Journal*, 44(2):427–440, 2005.

[2] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In *19th ACM Symposium on Operating Systems Principles (SOSP'03)*, The Sagamore, Bolton Landing (Lake George), NY, October 2003. ACM SIGOPS.

[3] P. Bohrer, R. Rajamony, and H. Shafi. Method and apparatus for accelerating Input/Output processing using cache injections, March 2004. US Patent No. US 6,711,650 B1.

[4] Ram Huggahalli, Ravi Iyer, and Scott Tetrick. 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.

[5] Edgar A. León and Michal Ostrowski. An infrastructure for network development. Proof of concept: Fast UDP. In *USENIX'05 Annual Technical Conference. Poster Session*, Anaheim, CA, April 2005.

[6] 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.

[7] Richard P. Martin, Amin M. Vahdat, David E. Culler, and Thomas E. Anderson. Effects of communication latency, overhead, and bandwidth in a cluster architecture. In *Proceedings of the 24th International Symposium on Computer Architecture (ISCA '97)*, pages 85–97, Denver, CO, June 1997.