Locality-aware Buffer Management: Algorithms Design and Systems Implementation for Data Intensive Applications

(A Brief Progress Report) *

Xiaodong Zhang

Department of Computer Science and Engineering
The Ohio State University
Columbus, OH 43210
zhang@cse.ohio-state.edu

Abstract

The speed gap between data processing in CPU and data accessing in disks has reached to an intolerable level, and will only become worse as time goes by. This bottleneck has seriously hindered the development of large scale computing systems for data-intensive applications that demand fast accesses to a huge amount of data. Viable and cost-effective solutions to address this problem are to build large memory buffers to cache data for reuse by taking advantage of low price and large capacity of DRAM memory, and to prefetch data for predicted future use by taking advantage of high and idle bandwidths of networks. We are working on four different research projects on locality-aware buffer management for data intensive applications. This report briefly presents the background, motivation, and working progress of our work funded by the NSF NGS program.

*This work is supported in part by the National Science Foundation under grant CNS-045909.

1 Introduction

With the rapid advancement of processor and networking technology, and with the falling price of memory and disks, computing resources of CPU cycles, available bandwidths at different levels of inter- and external connections (memory, I/O, and Internet), and large capacity of memory and disks are increasingly plentiful to us to build large systems. Many high performance systems have been built with thousands of CPUs and many large disks. For example, the ASCI Q cluster at the Los Alamos National Lab [1] has 2,048 HP ES45 nodes (8192 processors). Each user home directory is maintained in a central Network File System (NFS) servers with hundreds of disks. In order to satisfy increasingly high demand of accessing a huge amount of data for many users of scientists and application practitioners, several international efforts have been made to build globally distributed data centers. One such an example is the World-Wide Telescope (WWT) that is a virtual observatory facilitated by astronomy and astrophysics databases all over the world [13]. The goal of WWT is to allow scientists and students anywhere in the world to easily and quickly access the data for various

1-4244-0910-1/07/$20.00 ©2007 IEEE

purposes of research and education. Large scale scientific simulations and experiments can generate huge amounts of data for researchers all over the world to share and access. The source of the data is centrally stored in a mass storage system (MSS) that has extremely high latency of data retrieval, and an even longer time to transfer over a network. To provide fast data access service, multi-level memory buffers and disks are built between a client and the data source in MSS. Another example in data-intensive computing community is the international collaboration of high energy physicists on sharing a huge amount data in the world's most powerful particle accelerator at CERN, the European Organization for Nuclear Research, on the boarder between France and Switzerland near Geneva. The multi-level storage consists 5 levels from tier 0 (the data source), to tier 4 (the client's data storage) [2].

On the positive side, as long as we can feed the data to processors, data processing can be done by over-supplied CPU cycles. In addition, we are now able to store a huge amount of data in many disks with low prices. However, on the negative side, we are facing an increasingly more serious system problem: **moving data from huge storage space to very fast processors has become a bottleneck** for many data-intensive applications on large systems. The main reason behind this is that the improvement of data access latency, particularly, the access latency to disks, has been significantly lagged behind [12]. The computing performance has become increasingly dependent on the memory buffer due to the widening gap between processor speed and the disk access speed. A few years ago, a memory buffer should hold cached disk data if it is accessed again in next 5 minutes. Today, this benchmark of 5 minutes has increased to 35 minutes, and will soon reach to 1 hour and more [6]. In other words, the memory buffer must hold a large amount of data for programs and users to access in an increasingly long reuse time interval. Increasing the size of the memory buffer is only the trivial part of the solution, and the crucial issue is to develop efficient buffer management systems to adapt the dramatic technology changes and the high demand of data-intensive applications with complicated access patterns.

Although memory buffer management is a fundamental technology to improve the I/O performance, existing solutions in the running systems often fail to meet the needs of large scale applications. For example, a large system is normally built by a distributed infrastructure, such as clusters, multi-level data servers connected in local area networks, and data grids connected in global Internet. Existing memory buffer management on many complex and distributed systems still relies on simple LRU replacement mechanism, which causes low performance and inefficient usage of system resources. We have identified four different system problems and currently developing technical solutions to address these problems.

2 Locality-aware data replacement

Existing memory buffer management design and implementation lack efficient mechanisms to learn and adapt certain types of access patterns and locality behaviors of applications, causing low memory buffer hit ratios for some very important scientific and commercial applications, such as one-time access patterns, loop accesses, data accesses with long inter-reference accesses, and other accesses with weak locality. Simply increasing the size of the memory buffer would not solve the problem at all.

We have developed an effective buffer management algorithm called LIRS [9], which was well received in the system community. However, a disadvantage is that an implementation of LIRS in systems needs a global synchronization mechanism to serialize the operations in stacks/queues. This weakness hinders the impact of algorithmic efforts on real systems. We have made an effort to approximate LIRS by the Clock algorithm [5] that has been dominant in the operating systems for about 40 years. Clock-pro [7] is the result of our efforts. Since a clock-based algorithm is synchronization-free, it is scalable and is a realistic solution in real

systems. The Linux development community has developed two versions of Clock-pro patches for runtime execution [4].

3 Buffer management in multi-level caching systems

In a large distributed systems, the data are cached in multi-level memory buffers connected by fast clusters or networks, data access patterns and locality strengths are different at different levels, adapting data access behaviors in multiple memory buffers is even more difficult and challenging. Most existing buffer management in distributed systems simply adopt the mechanism used in a single-level memory buffer system, degrading the memory caching performance even more seriously than that in a single-level buffer. One major problem is that the locality information embedded in the streams of access requests from clients is not consistently analyzed and exploited, resulting in globally nonsystematic, and therefore suboptimal, placement and replacement of cached blocks across the hierarchy. We have proposed a coordinated multilevel cache management protocol based on consistent access-locality quantification and show its effectiveness in different platforms [10, 11].

4 Putting the Disk Spatial Locality Information onto the OS Map

For a given amount of data, sequential accesses are several orders of magnitude faster than random accesses in a disk. Unfortunately efforts of organizing sequential disk accesses have been largely ignored in the memory buffer management. This is because memory buffer can effectively observe the access behavior of applications, but has no knowledge about data placements in disks. We have made efforts to put the disk spatial locality information on the operating system map so that the buffer management can exploit both temporal locality based on program execution and spatial locality based on data placements in disks [8].

5 Energy-efficient disk operations in mobile computers

In a mobile computer the hard disk consumes a considerable amount of energy. Existing disk dynamic power management policies usually take conservative approaches to save energy, and disk energy consumption remains a serious issue. Recently the flash drive is becoming a must-have portable storage device for almost every laptop user on travel. We have made efforts to make another highly desired use of the flash drive — saving disk energy. This is achieved by using unused space in the flash drive as a standby buffer for caching and prefetching disk data. Our design can significantly extend disk idle time with careful and deliberate consideration of unique characteristics of the flash drive. The energy saving solution is called "Smart-Saver" [3].

6 Acknowledgment

Improving I/O performance has been an active research in our group. Song Jiang (a former Ph.D. student, and currently an assistant professor at the Wayne State University), has made strong efforts and contributions to this research. Other former and current Ph.D. students have made best efforts to several related research projects: Feng Chen (Ohio State), Songqing Chen (assistant professor, George Mason University), Xiaoning Ding Ohio State), Lei Guo (Ohio State), Shuang Liang (Ohio State), Shansi Ren (Ohio State), and Enhua Tan (Ohio State).

References

[1] ASCI-Q machine, http://www.lanl.gov/asci/.

[2] CERN: European organization for Nuclear Research, http://cern.web.cern.ch/CERN/.

[3] F. Chen, S. Jiang, and X. Zhang, "SmartSaver: turning flash drive into a disk energy saver for mobile computers", *Proceedings of the 11th International Symposium on Low*

Power Electronics and Design (ISLPED'06), Tegernsee, Germany, October, 2006.

[4] Clock-pro in Linux kernels, http://linux-mm.org/ClockProApproximation.

[5] F. J. Korbato. "A paging experiment with the multics system", *MIT Project MAC Report MAC-M-384*, May 1968.

[6] J. Gray. "Greetings! from a file system user", *Keynote Speech in 2005 USENIX Conference on File and Storage Technologies* (FAST'05), San Francisco, CA, December 2005.

[7] S. Jiang, F. Chen, and X. Zhang, "CLOCK-Pro: an effective improvement of the CLOCK replacement", *Proceedings of USENIX Annual Technical Conference* (USENIX'05), Anaheim, CA, April 200, pp. 223-236.

[8] S. Jiang, X. Ding, F. Chen, E. Tan, and X. Zhang, "DULO: an effective buffer cache management scheme to exploit both temporal and spatial localities", *Proceedings of the 4th USENIX Conference on File and Storage Technologies* (FAST'05), San Fransisco, CA, December 2005, pp. 101-114.

[9] S. Jiang and X. Zhang, "LIRS: an efficient low inter-reference recency set replacement to improve buffer cache performance", *Proceedings of 2002 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, Marina Del Rey, CA, June, 2002, pp. 31-42.

[10] S. Jiang and X. Zhang, "ULC: a file block placement and replacement protocol to effectively exploit hierarchical locality in multi-level buffer caches", *Proceedings of the 24th International Conference on Distributed Computing Systems* (ICDCS'04), Tokyo, Japan, March 2004, pp. 168-177.

[11] S. Jiang, K. Davis, and X. Zhang, "Coordinated multilevel buffer cache management with consistent access locality qualification", *IEEE Transactions on Computers*, Vol. 56, No. 1,, 2007, pp. 95-108.

[12] D. A. Patterson, "Latency lags bandwidth", *Communications of the ACM*, Vol. 47, No. 10, October 2005, pp. 71-75.

[13] A. S. Szalay and J. Gray, "The world-wide telescope", *Science*, volume 293 (5537), September 2001, pp. 2037-2040.