UCRL-ID-153361

Memory System Technologies for Future High-End Computing Systems

S. A. McKee, B. R. de Supinski, F. Mueller, G. S. Tyson

U.S. Department of Energy

Lawrence Livermore National Laboratory

May 16, 2003

This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

Memory System Technologies for Future High-End Computing Systems

Sally A. McKee (Cornell Univ.) Bronis R. de Supinski (LLNL)
Frank Mueller (North Carolina State Univ.)
Gary S. Tyson (Univ. of Michigan)

May 16, 2003

A Introduction

Our ability to solve Grand Challenge Problems in computing hinges on the development of reliable and efficient High-End Computing systems. Unfortunately, the increasing gap between memory and processor speeds remains one of the major bottlenecks in modern architectures. Uniprocessor nodes still suffer, but symmetric multiprocessor nodes — where access to physical memory is shared among all processors — are among the hardest hit. In the latter case, the memory system must juggle multiple working sets and maintain memory coherence, on top of simply responding to access requests. To illustrate the severity of the *current* situation, consider two important examples: even the high-performance parallel supercomputers in use at Department of Energy National labs observe single-processor utilization rates as low as 5%, and transaction processing commercial workloads see utilizations of at most about 33%. A wealth of research demonstrates that traditional memory systems are incapable of bridging the processor/memory performance gap, and the problem continues to grow. The success of future High-End Computing platforms therefore depends on our developing hardware and software technologies to dramatically relieve the memory bottleneck.

In order to take better advantage of the tremendous computing power of modern microprocessors and future High-End systems, we consider it crucial to develop the hardware for intelligent, adaptable memory systems; the middleware and OS modifications to manage them; and the compiler technology and performance tools to exploit them. Taken together, these will provide the foundations for meeting the requirements of future generations of performance-critical, parallel systems based on either uniprocessor or SMP nodes (including PIM organizations). We feel that such solutions should not be vendor-specific, but should

be sufficiently general and adaptable such that the technologies could be leveraged by any commercial vendor of High-End Computing systems. This strategy is likely to have the most impact while maintaining modest costs for adoption of the new technologies.

B Hardware and Software Technologies

B.1 Key Technologies

Many underlying pieces of the necessary technologies have been developed within the last decade. The Stream Memory Controller project [9] demonstrated the viability and performance impact (speedups of up to $23x$) of putting intelligent DRAM resource management and compiler-controlled prefetching into the memory controller. The Impulse Adaptable Memory Controller project [14] built on these techniques, and contributed technology to perform parallel vector element accesses (for strided or indirection-vector access patterns) in the DRAM backend [8], as well as the ability to remap physical addresses to dramatically improve utilization of the cache hierarchy. Initial compiler technology has been developed to exploit such memory systems [1, 6], and techniques ranging from traditional loop and data restructuring optimizations [10, 3] to data shackling [7] and computation regrouping for temporal locality [12] augment the impact of these DAOs.

Processor-in-Memory (PIM) technology [13, 5] will dramatically change the location of some system bottlenecks, but regardless of the memory organization, it will always be essential to exploit available resources intelligently if we are to realize the highest possible system performance. Likewise, sophisticated interconnect fabrics like Myrinet [2, 11] enable a host of performance optimizations, but intelligent memory systems can be used to further exploit these capabilities, e.g., by performing scatter/gather operations directly to/from the NIC.

B.2 Areas for Development

There exist two show-stopper problems to be solved before we can realize the full benefit of these technologies: *memory coherence*, and *software automation*.

Current systems conservatively apply cache invalidations and usage restrictions, avoiding the creation of aliased copies in caches, or leaving the responsibility for coherence management to the user. This greatly reduces system efficiency and applicability. Further, existing solutions are largely limited to uniprocessor systems. Solving the general problem of memory aliases for reconfigurable parallel systems represents one of the key problems faced by high-performance computer architects today, and no comprehensive, general solution has yet been developed. Likewise, even though proof-of-concept compiler prototypes or algorithms have been developed for these sophisticated memory systems, even better analysis and tools

are needed to take full advantage of optimization opportunities without requiring significant programmer intervention.

Efficiently maintaining coherence is a primary requirement in any parallel memory system. The problem of enforcing an application's required level of coherence becomes much more complex in memory systems that provide processors with multiple or reconfigurable views of application data. For instance, potential problems arise from the memory remappings of the Impulse Adaptable Memory System; the reconfigurable, application-specific hardware assists in the MORPH architecture [15]; the use of explicitly controlled scratchpad or streaming memory in the Malleable Cache project [4] or the non-cached stream buffers in the Stream Memory Controller; and even the E-registers of the Cray T3E. All these mechanisms enable memory aliases, whereby the same physical data element may be accessed using more than one address. This inevitably leads to coherence conflicts between stored copies of aliased memory elements.

C Technology Maturity Roadmaps, Investments, and Performance Metrics

As feature sizes shrink and microarchitectures continue to evolve, the processor/memory performance gap will continue to grow for High-End Systems. Even PIM systems will require off-chip memory storage for scientific and transaction processing codes (to name but a few examples) with huge datasets. The technologies outlined in Section B.1 will continue to mature, but the problems outlined in Section B.2 will not disappear. By addressing these problems in a structured, general, and flexible manner, we envision an avenue to improving the memory performance for traditional as well as revolutionary sytem designs. Deploying these solutions will represent but a marginal increase in hardware costs. Comprehensive compilation, dynamic optimization, and performance analysis tools will minimize the impact on legacy codes, and on source program structure, in general.

Developing and prototyping the hardware and software for the proposed advanced memory systems constitutes a four-five year time investment. The hardware component requires architectural design-space analysis and performance simulation, HDL synthesis for complexity and critical-path analyses, FPGA prototyping of functionality and interfaces, and technology transfer to generate a full system. The software components will proceed in parallel. Vendor-specific solutions involving technology transfer and collaboration from the beginning are likely to have a shorter project life cycle, but such an approach obstructs our goal of developing a general and widely applicable set of solutions. Performance metrics will include cycle-accurate simulation of current benchmarks from the SPEC 2000, ASCI

Purple and DARPA Data Intensive Systems suites, TPC-C, as well as emerging applications and traces from commercial (e.g., IBM DB2) transaction processing systems. Participating researchers from National Laboratories will have access to a wider variety of real codes with which to evaluate our solutions. Technologies to speed simulation times without distorting performance statistics will be key to the software infrastructure of the project, as will integrating hardware emulation and prototype component testing with flexible simulation of other parts of the large, High-End systems we target.

Current hardware trends emphasize the increasing importance of memory system performance. However, many current and emerging mechanisms, such as prefetching and speculative memory accesses, increase bandwidth requirements in order to hide latency. Other mechanisms, such as system-on-a-chip (SOC), increase the available bandwidth but do not address how to use that bandwidth effectively. Our solutions will complement these trends by reducing wasted bandwidth. Other benefits of our approach include moving important features of vector systems into COTS, and exploiting all memory resources (processor caches, PIM, traditional DRAM, and specialized buffers within the memory controller or network interface, along with the buses connecting these structures to other parts of the system) as fully as possible, both for uniprocessor and SMP nodes in very large scale parallel systems.

D Technology Dependencies and Interactions

Our solutions do not depend on the development of any particular foundation technologies, but will adapt to address emerging technologies, including PIM organizations, fast interconnects, SOC and CMP organizations, new DRAM interfaces, advances in FPGA and ASIC technologies, and the incorporation of multiple memory controllers within a single node. We envision a componentized memory system — a trend that is already appearing in high-performance machines such as the newest products from AMD. Organizing the memory system as a series of individual components both isolates complexity (simplifying design and test while minimizing hardware costs and unwanted microarchitectural interactions) and increases efficiency (by allowing components to be placed to minimize communication delays).

References

[1] M. Benitez and J. Davidson. Code generation for streaming: An Access/Execute mechanism. In *Proceedings of the 4th Symposium on Architectural Support for Programming Languages and Operating Systems*, pages 132–141, Apr. 1991.

[2] N. Boden, D. Cohen, R. Felderman, A. Kulawik, C. Seitz, J. Seizovic, and W.-K. Su. Myrinet – A gigabit-per-second local-area network. *IEEE MICRO*, 15(1):29–36, Feb. 1995.

[3] B. Chandramouli, J. B. Carter, W. C. Hsieh, and S. A. McKee. A cost framework for evaluating integrated restructuring optimizations. In *Proceedings of the 2001 International Conference on Parallel Architectures and Compilation Techniques*, Sept. 2001.

[4] D. Chiou, P. Jain, S. Devadas, and L. Rudolph. Dynamic cache partitioning via columnization. In *Proceedings of the 37th Conference on Design Automation*, June 2000.

[5] M. Hall, P. Kogge, J. Koller, P. Diniz, J. Chame, J. Draper, J. LaCoss, J. Granacki, J. Brockman, A. Srivastava, W. Athas, and V. Freeh. Mapping irregular appilcations to DIVA, a PIM-based data-intensive architecture. In *Supercomputing'99*, Nov. 1999.

[6] X. Huang, Z. Wang, and K. McKinley. Compiling for an impulse memory controller. In *Proceedings of the 2001 International Conference on Parallel Architectures and Compilation Techniques*, pages 141–150, Sept. 2001.

[7] I. Kodukula and K. Pingali. Data-centric transformations for locality enhancement. *International Journal of Parallel Programming*, 29(3):319–364, June 2001.

[8] B. Mathew, S. McKee, J. Carter, and A. Davis. Design of a parallel vector access unit for SDRAM memories. In *Proceedings of the Sixth Annual Symposium on High Performance Computer Architecture*, pages 39–48, Jan. 2000.

[9] S. McKee, W. Wulf, J. Aylor, R. Klenke, M. Salinas, S. Hong, and D. Weikle. Dynamic access ordering for streamed computations. *IEEE Transactions on Computers*, 49(11):1255–1271, Nov. 2000.

[10] K. S. McKinley, S. Carr, and C.-W. Tseng. Improving data locality with loop transformations. *ACM Transactions on Programming Languages and Systems*, 18(4):424–453, July 1996.

[11] S. Paikin, Lauria, and A. Chien. High performance messaging on workstations: Illinois fast messages (fm) for myrinet. In *Proceedings of Supercomputing '88*, 1995.

[12] V. Pingali, S. McKee, W. Hsieh, and J. Carter. Computation regrouping: Restructuring programs for temporal data cache locality. In *Proceedings of the 2002 International Conference on Supercomputing*, pages 252–261, June 2002.

[13] T. Sunaga, P. M. Kogge, et al. A processor in memory chip for massively parallel embedded applicatiions. *IEEE Journal of Solid State Circuits*, pages 1556–1559, Oct. 1996.

[14] L. Zhang, Z. Fang, M. Parker, B. Mathew, L. Schaelicke, J. Carter, W. Hsieh, and S. McKee. The impulse memory controller. *IEEE Transactions on Computers*, 50(11):1117–1132, Nov. 2001.

[15] X. Zhang, A. Dasdan, M. Schulz, R. Gupta, and A. Chien. Architectural adaptation for application-specific locality optimizations. In *Proceedings of the 1997 IEEE International Conference on Computer Design*, 1997.