Implications of Memory Interference for Composed HPC Applications

Brian Kocoloski, Yuyu Zhou, Bruce Childers, and John Lange
Department of Computer Science
University of Pittsburgh
{briankoco,yuyuzhou,childers,jacklange}@cs.pitt.edu

ABSTRACT

The cost of inter-node I/O and data movement is becoming increasingly prohibitive for large scale High Performance Computing (HPC) applications. This trend is leading to the emergence of composed *in situ* applications that co-locate multiple components on the same node. However, these components may contend for underlying memory system resources. In this extended research abstract, we present a preliminary evaluation of the impacts of contention for shared resources in the memory hierarchy, including the last level cache (LLC) and DRAM bandwidth. We show that even modest levels of memory contention can have substantial performance implications for some benchmarks, and argue for a cross layer approach to resource partitioning and scheduling on future HPC systems.

CCS Concepts

•General and reference → Design; •Software and its engineering → Memory management; Ultra-large-scale systems;

Keywords

High Performance Computing; Composed Applications; Shared Memory

1. INTRODUCTION

Tightly-coupled and massively parallel High Performance Computing (HPC) applications have historically been afforded the luxury of systems devoted entirely to the forward progress of a single application at a time. However, while it is common to decompose today's large scale applications onto separate nodes where the components (e.g., scientific simulation, data post processing, visualization of results) communicate over an interconnect, emerging workloads, such as *in situ* applications, are being developed to allow components to communicate more efficiently over local inter-process communication (IPC) resources (e.g., shared memory) [12]. Such consolidation is considered to be a useful substrate to reduce the prohibitive performance and power costs that would accompany inter-node I/O and data movement on extreme scale systems. [4].

Although consolidation may provide numerous benefits, it introduces the need for resource sharing to HPC applications that heretofore have largely operated in isolated environments. This sharing will likely have performance implications across the entire system stack, where core OS services, device drivers and network stacks, and the underlying hardware will all be shared to some degree by separate application components. In this extended research abstract, our focus is the implications of sharing resources in the memory system. We present a preliminary experimental analysis of a set of HPC benchmarks configured to operate alongside a co-running benchmark that consumes various degrees of last-level cache (LLC) capacity as well as memory (DRAM) bandwidth. Our evaluation demonstrates that modest contention for these resources can have a considerable impact on HPC application performance. Furthermore, we propose that cross layer design decisions based on the prioritization of performance sensitive HPC simulations will be required to support future composed HPC workflows.

2. HPC CONSOLIDATION CHALLENGES

Consolidation of mixed workloads has long been a hallmark of shared commodity computing systems, mainly because it provides the ability to maximize system throughput using strategies based on resource sharing, while providing performance guarantees and isolation as secondary design goals. However, the advent of multi-/many- core processors and increasing amounts of per node memory has increased the amount of consolidation, and thus, the degree of sharing in the underlying architecture. As a result, resource contention has emerged as a significant problem [2]. Recent work has investigated strategies for managing resources in the memory system and improving performance through techniques such as memory channel partitioning [7] and DRAM bank-aware memory allocation [11]. Additionally, a plethora of research has approached the problem from the standpoint of maximizing system throughput while maintaining quality of service (QoS). Such techniques include QoS-aware DRAM scheduling [8] and virtual machine migration [6] based on measured QoS violations.

While these and many other approaches address the issues of resource contention in shared systems, a distinguishing characteristic of emerging HPC applications is that strict performance isolation is far more important than the maximization of overall system throughput. Thus, though there has been an abundance of research enabling effective consolidation of shared systems, these approaches are generally built to optimize objectives (maximize system throughput while meeting QoS) that are not well-aligned with those of composed applications (maximize performance of the simulation by providing direct and predictable access to hardware).

MEMSYS '15 October 05-08, 2015, Washington DC, DC, USA

ACM ISBN 978-1-4503-3604-8/15/10.

DOI: http://dx.doi.org/2818950.2818965

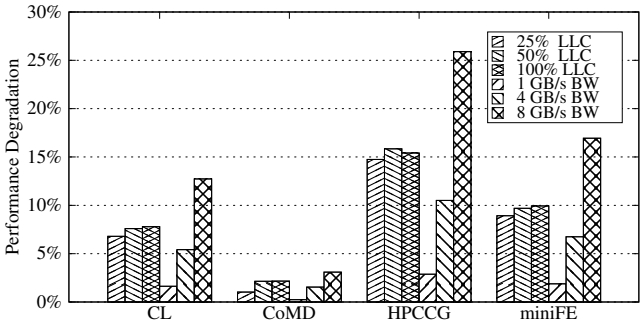

Figure 1: Impact of contention for shared resources in the memory hierarchy

STREAM (GB/s)	System-Wide DRAM Accesses (GB/s)			
	CL	CoMD	HPCCG	miniFE
0	19.96	2.64	26.59	26.48
1	20.84	2.86	26.91	26.87
4	22.09	6.51	27.54	27.54
8	23.84	11.11	27.88	27.81

Table 1: Maximum DRAM bandwidth delivered for different workloads

Thus, it is our position that this emerging problem space will require a new focus on the prioritization of individual HPC workflow components that are most sensitive to contention in the memory system, and should be given priority to shared resources over less sensitive co-located components that can be executed on a "best effort" basis. We envision that potential solutions will necessarily span a large portion of the system stack, including explicitly stated application requirements, operating system (OS) task mapping and scheduling decisions based on these explicit requirements, and finally the fine-grained partitioning and dedication of resources in the memory hierarchy.

3. PRELIMINARY EVALUATION

Our preliminary evaluation was designed to measure the performance implications of sharing memory resources between multiple components of a composed workflow. We selected a set of HPC benchmarks from the Mantevo Project [1] as a traditional HPC simulation, and used the STREAM benchmark [2] as a co-running component due to the ease with which it can be configured to consume various levels of shared resources. The experiments were performed on a single-socket 6-core Intel Xeon processor running at 2.10 GHz. The socket had 12 GB DRAM, per-core 32 KB L1 I&D caches, a 1.5 MB L2 cache, and a 15 MB L3 cache (LLC). The HPC simulation ran on 4 cores, while STREAM ran on 1 core. To measure the impact of LLC contention, we limited STREAM's working set size to consume 25%, 50%, and 100% of the LLC. To measure the effects of bandwidth contention, we increased the working set and throttled STREAM's execution rate to consume 1 GB, 4 GB, or 8 GB of DRAM bandwidth per second, out of the nearly 28 GB/s peak DRAM bandwidth supported by the socket. [3]

Figure 1 shows the results of our LLC contention experiments. The figure demonstrates that even when STREAM's working set is limited to only 25% of the LLC, 3 of the 4 HPC benchmarks experience slowdowns of at least 7% and up to 15%. Interestingly, the figure also shows that increasing the working set size to consume 50% or even 100% of the LLC does not cause much further degradation to the HPC benchmarks. This suggests that LLC capacity is a critical resource for these benchmarks, to the degree that a modest level of contention can maximize the impact of pollution.

The results of the bandwidth contention experiments are shown in Figure 1 and Table 1. When STREAM requires only 1 GB/s of bandwidth, each benchmark experiences less than 3% overhead. However, increasing the bandwidth usage to 4 GB/s and then to 8 GB/s causes up to a 25% performance reduction (HPCCG). As Table 1 shows, for HPCCG and miniFE in particular, this overhead can be partly attributed to the fact that the system is nearing its peak DRAM bandwidth. The top row indicates that these two benchmarks are capable of nearly consuming the full DRAM bandwidth by themselves, and thus adding contention via STREAM has the effect of pushing the limits of the memory system. However, the overheads are also experienced, albeit to lesser degrees, by the CL and CoMD benchmarks, which exhibit the same trends in performance reduction as bandwidth increases. We attribute this overhead largely to the increased latency of the DRAM accesses caused by queuing in the memory controller.

4. DISCUSSION AND FUTURE WORK

Our preliminary evaluation implies that future work will be needed to ensure that performance sensitive HPC simulations are not perturbed by their co-running components. Technologies such as 3D stacked memory (e.g., hybrid memory cubes [10]) are emerging, and these architectures may alleviate bandwidth contention to some degree. However, future systems are likely to consist of processors with increasingly high core counts, as well as heterogeneous (co)processors (e.g. GPUs, AMD APUs, Intel Xeon Phis, etc.) that will have significant bandwidth requirements, and thus it is unlikely that bandwidth will become a "free" resource.

We propose that cross layer techniques based on explicit bandwidth partitions could allow composed HPC workflows to state their requirements to the system, thus allowing the prioritization of performance sensitive workflow components. We envision that such techniques would ideally be supported directly in hardware as well as the application runtimes and OSes, although it is possible that the OS alone could effectively allocate bandwidth by monitoring memory access patterns and enforcing bandwidth constraints through intelligent scheduling decisions (e.g., throttling). Furthermore, enabling component-level partitioning of LLC capacity and memory channels could allow for stricter performance guarantees to be made, although they may reduce performance in the common case by reducing memory-level parallelism.

We plan to leverage our experience as part of the Hobbes project [1] for investigating the feasibility of such techniques. Hobbes is a Department of Energy supported project aiming to develop an operating system and runtime for future extreme scale systems. Part of the Hobbes motivation is that commodity consolidation techniques are not well aligned with the requirements for composition of HPC workflows. Recent Hobbes work has targeted better support for composition through the deployment of specialized OS kernels that can be designed to execute a target workflow component [9, 3]. Other Hobbes work has investigated scheduling mechanisms to facilitate consolidation of components on the same CPU cores [5]. In future work, we plan to investigate how Hobbes can facilitate the fine-grained partitioning and sharing of resources in the memory hierarchy as required by these workflows.

[1] https://mantevo.org

[2] http://www.cs.virginia.edu/stream/

[3] Memory bandwidth was measured using the Intel PCM tool: https://software.intel.com/en-us/articles/intel-performance-counter-monitor

Finally, it is also important to consider the performance of real *in situ* and other composed workflows that share memory in order to accurately gauge the level of contention experienced in these applications. While many in the HPC community agree that consolidation is likely to be prevalent in future systems, the opinion is not universally shared, particularly due to the demonstrated challenges associated with resource sharing in these applications. As such, given that the implications for many target proxy applications/drivers have not been well studied in consolidated systems, simply characterizing these workflows, and determining which types of applications are more amenable to consolidation than others will likely add value to the HPC community.

5. CONCLUSION

Emerging HPC workflows are likely to require the consolidation of multiple workloads on the same shared infrastructure. This extended abstract evaluated the impacts of contention for shared resources on a set of HPC benchmarks and showed that both LLC pollution and memory bandwidth contention caused by consolidation lead to substantial performance interference (up to 15% and 25%, respectively). We propose that cross layer techniques, ranging from application-level specification of resource requirements, to OS-level enabling of direct hardware access, and finally to hardware-level partitioning of resources will be needed to fully support consolidation in future HPC systems.

6. REFERENCES

[1] R. Brightwell et al. Hobbes: Composition and Virtualization as the Foundations of an Extreme-Scale OS/R. In *Proc. 3rd International Workshop on Runtime and Operating Systems for Supercomputers (ROSS)*, 2013.

[2] J. Dean and L. A. Barroso. The Tail at Scale. *Communications of the ACM*, 56(2), 2013.

[3] K. Hale and P. Dinda. A Case for Transforming Parallel Run-times into Operating System Kernels. In *Proc. 24th International ACM Symposium on High Performance Parallel and Distributed Computing (HPDC)*, 2015.

[4] P. Kogge et al. Exascale Computing Study: Technology Challenges in Achieving Exascale Systems. Technical report, University of Notre Dame CSE Department, 2008.

[5] O. Mondragon et al. Quantifying Scheduling Challenges for Exascale System Software. In *Proc. 5th International Workshop on Runtime and Operating Systems for Supercomputers (ROSS)*, 2015.

[6] J. Moses et al. Shared Resource Monitoring and Throughput Optimization in Cloud-Computing Datacenters. In *Proc. 25th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2011.

[7] S. Muralidhara et al. Reducing Memory Interference in Multicore Systems via Application-Aware Memory Channel Partitioning. In *Proc. 44th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2011.

[8] O. Mutlu and T. Moscibroda. Parallelism-Aware Batch Scheduling: Enhancing both Performance and Fairness of Shared DRAM Systems. In *Proc. 35th International Symposium on Computer Architecture (ISCA)*, 2008.

[9] J. Ouyang et al. Achieving Performance Isolation with Lightweight Co-Kernels. In *Proc. 24th International ACM Symposium on High Performance Parallel and Distributed Computing (HPDC)*, 2015.

[10] J. T. Pawlowski. Hybrid Memory Cube (HMC). In *Hot Chips (HC23)*, 2011.

[11] H. Yun et al. PALLOC: DRAM Bank-Aware Memory Allocator for Performance Isolation on Multicore Platforms. In *Proc. 2oth IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2014.

[12] F. Zheng et al. GoldRush: Resource Efficient In Situ Scientific Data Analytics Using Fine-Grained Interference Aware Execution. In *Proc. 26th. International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2013.