

Virtualization Aspects of High Performance Computing (Extended Abstract) *

Lamia Youseff

Rich Wolski

Department of Computer Science, University of California, Santa Barbara.
{lyouseff, rich}@cs.ucsb.edu

Abstract

Our research has revealed that paravirtualization imposes minimal performance on high performance computing workloads, while exposing numerous benefits for this field. In this extended abstract, we overview the work we have done in investigating the performance ramifications of deploying paravirtualized system in HPC setting. Furthermore, we overview our results of characterizing paravirtualized memory hierarchy, performance boundary, and paravirtualization impact on autotuning linear algebra software systems. The next step in our research is to support efficient memory sharing, synchronization and communication scheme between applications running in distinct virtual machines to enable efficient execution between threads in HPC applications.

1. Keywords

Operating System, Virtualization, High Performance Computing, Performance Analysis.

2. Introduction

Historically, virtualization has been ignored in computational intensive settings as a result of its potential for performance retardation. *Paravirtualization*, however offers an attractive alternative as the guest and the host OSes are both strategically modified to provide optimized performance. To this end, our research and others [5, 3, 4] have measured the performance ramifications of running general HPC benchmarks on paravirtualized systems. At the same time, other studies

have focused on the flexibility and functionality benefits of using paravirtualization in HPC. For example, OS-level check-pointing, fault-tolerance and load balancing are very attractive possibilities. In addition, we and other researchers have looked into dynamic adaptation of the guest OS and application-customized guest OSes for scientific parallel codes [1, 2].

In this extended abstract, we present an overview of our research addressing the performance analysis of executing HPC workloads on paravirtualized systems as well as our current research direction.

3. Paravirtualization Performance in HPC

A comprehensive performance evaluation of Xen, a low-overhead, Linux-based, virtual machine monitor, for paravirtualization of HPC cluster systems was part of our research goals. For that, we investigate individual subsystems and the overall system performance using a wide range of benchmarks and applications. In addition, we compare the performance of a paravirtualized kernel against Red Hat Enterprise 4 Linux (RHE) and the LLNL's *Clustered High Availability OS* (CHAOS) kernel, using *t-test* at the $\alpha \geq 0.95$ significance level. Our results indicate that Xen is very efficient and practical for HPC systems. As a representative of our results, we present here the results from the NAS Parallel Benchmarks (NPB).

NAS parallel benchmarks (NPB) mimic the computational, communicational and data movement characteristics of large scale computational fluid dynamics applications. Figure 1 depicts the performance of several NPB codes. We observe that all of the kernels perform similarly for Embarrassingly Parallel (EP), Integer Sort (IS), Linear Solver (LU) and Multi Grid (MG) codes. This is interesting since the benchmarks

*Sponsored in part by NSF grants (ST-HEC-0444412 and CCF-0331645), this abstract is a summary of our research on paravirtualization for high performance computing. Published papers of our research are available at <http://www.cs.ucsb.edu/~lyouseff>

are very different in terms of their behavior: EP performs distributed computation with little communication overhead, IS performs a significant amount of communication using collective operations, and MG employs a large number of blocking send operations.

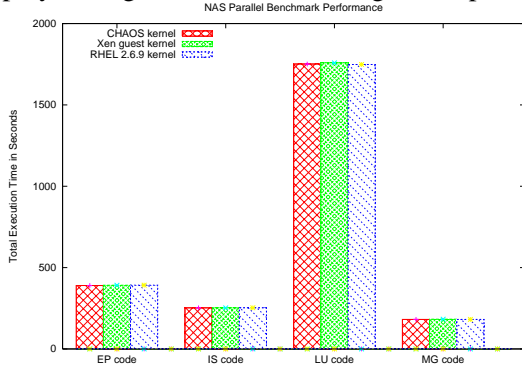


Figure 1. shows the NPB codes performance.

In addition, we investigate the impact of paravirtualization on the different levels of the memory hierarchy. Towards this end, we explore the paravirtualized memory hierarchy behavior using a double-precision matrix matrix multiplication code that uses the *BLAS* level-3 libraries as a driver code for our experiments, which is characterized by a growing memory consumption of up to 350 MB.

We show the results of our experiments in figure 2. Each of the two subfigures plots the performance in Mflops on the y -axis as a function of the matrix dimension on the x -axis, under two different main-memory configurations. Although the memory size did not impact the Mflops achieved by the DGEMM driver code up to matrix dimension of 3100, its impact was clearly encountered by larger matrices running on limited main-memory OS kernels. Further investigation of the swapping activity, TLB performance and memory management and evacuation policy is included in our publications. In addition, we found that there is no significant difference in performance between the native and paravirtualized execution even when *ATLAS*, a widely used autotuning software for linear algebra libraries, tunes the performance of the libraries to near peak speeds. The results show that the combination of *ATLAS* autotuning and Xen paravirtualization deliver native execution, peak performance and nearly identical memory hierarchy performance profiles.

The next step in our research is to allow different lightweight OS instances, as well as heavier-

weight utility operating systems coexisting on the same physical multi-core machine, to cooperate simultaneously, under the control of one application. To that end, we are advocating the unavoidable necessity of transparent and flexible efficient memory sharing and synchronization across light-weight OS instances. To demonstrate our assertion, our primary investigation present an evaluation of the different current inter-virtual machine communication themes, and evaluate their performance ramifications and limitations. It shows that it is possible to enable native execution speeds with efficient memory sharing mechanism. This, in turn will allow efficient communication patterns between virtual machine in heterogenous large scale machine and peta-scale computing infrastructures.

4. Conclusion

Our results illustrate that paravirtualization has no significant impact on the performance of HPC workloads in general, and on memory-intensive applications in specific, even when memory becomes a scarce resource. Paravirtualization, furthermore, does not alter the system image and does not affect the ability of empirically tuned codes to produce peak performance for linear algebra software. Given the rise of the new paradigm of cloud computing, paravirtualization exposes new deployment scenarios for linear algebra computational kernels and software.

References

- [1] G. B. et al. Application-Specific Customization on Many-Core Platforms: The VT-ASOS Framework. In *Proceedings of the Second Workshop on Software and Tools for Multi-Core Systems*, March 2007.
- [2] L. Y. et al. Linux Kernel Specialization for Scientific Application Performance. Technical Report UCSB Technical Report 2005-29, Univ. of California, Santa Barbara, Nov 2005.
- [3] L. Y. et al. Evaluating the Performance Impact of Xen on MPI and Process Execution For HPC Systems. In *VTDC '06*, 2006.
- [4] L. Y. et al. Paravirtualization for HPC Systems. In G. M. et al., editor, *ISPA Workshops*, volume 4331 of *Lecture Notes in Computer Science*, pages 474–486. Springer, 2006.
- [5] M. e. a. Mergen. Virtualization for High-Performance Computing. *SIGOPS Oper. Syst. Rev.*, 40(2):8–11, April 2006.

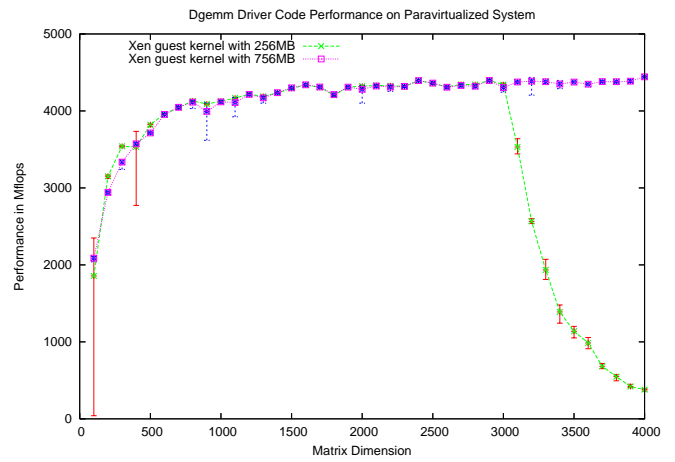
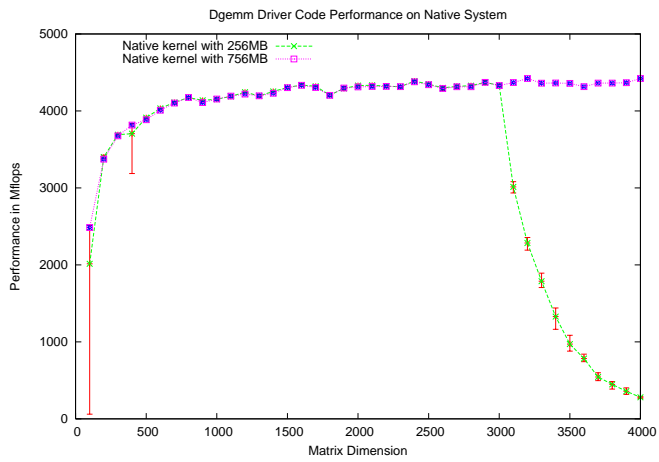


Figure 2. The performance of the DGEMM driver code as a function of the matrix dimensions.