The Effect of Multi-core on HPC Applications in Virtualized Systems

Jaeung Han[1], Jeongseob Ahn[1], Changdae Kim[1], Youngjin Kwon[1], Young-ri Choi[2], and Jaehyuk Huh[1]

[1] Computer Science, KAIST, Daejeon, Korea
[2] Korea Institute of Science and Technology Information (KISTI), Daejeon, Korea

Abstract. In this paper, we evaluate the overheads of virtualization in commercial multicore architectures with shared memory and MPI-based applications. We find that the non-uniformity of memory latencies affects the performance of virtualized systems significantly. Due to the lack of support for non-uniform memory access (NUMA) in the Xen hypervisor, shared memory applications suffer from a significant performance degradation by virtualization. MPI-based applications show more resilience on sub-optimal NUMA memory allocation and virtual machine (VM) scheduling. However, using multiple VMs on a physical system for the same instance of MPI applications may adversely affect the overall performance, by increasing I/O operations through the domain 0 VM. As the number of cores increases on a chip, the cache hierarchy and external memory will become more asymmetric. As such non-uniformity in memory systems increases, NUMA and cache awareness in VM scheduling will be critical for shared memory applications.

1 Introduction

Virtualization has become popular to improve system utilization by consolidating multiple servers into a physical system. In addition to the improved utilization, other benefits of virtualization, such as flexible resource management, fault isolation, and support for different operating systems, have led to the increase of interest in the virtualization of computing clusters for high performance computing (HPC). Furthermore, public cloud computing services, such as Amazon EC2 [1], accelerated the adoption of virtualization for HPC applications.

However, the characteristics of compute-intensive HPC applications are quite different from those of I/O-intensive server applications, which have been the main target of prior performance optimizations for virtualization. To adopt virtualization for HPC applications, thorough analysis of their performance characteristics in virtualized systems is necessary. Furthermore, the fast increase of core counts in multicore architectures, combined with virtualization techniques, affects the performance of HPC applications significantly.

In multicore architectures, memory hierarchies are getting complicated, and their effects have become significant for HPC applications. One of the most important factors in multicore memory hierarchies is non-uniform memory access

(NUMA). Virtualization complicates the scheduler optimizations for NUMA, as it hides the underlying non-uniformity in memory access. In virtualized systems, a guest operating system may not be aware of the non-uniformity in memory access, and thus it may not be able to make optimal scheduling decisions.

In this paper, we investigate the overheads of virtualization on HPC applications running on multicore systems with uniform and non-uniform memory access latencies. Using the Xen hypervisor, we evaluate both a shared-memory multi-threaded benchmark, PARSEC [4], and a MPI-based benchmark, NAS Parallel Benchmark (NPB) [3] in various configurations. The experimental results show that for shared memory applications, the performance overheads by virtualization are minor, if all the cores have a uniform memory latency. However, if the memory hierarchy is not symmetric, especially with non-uniform memory access times, the current Xen hypervisor [6] adds a significant overhead to the applications by sub-optimal scheduling and memory allocation.

For MPI-based applications, the impact of non-uniform memory latencies is not as severe as the shared-memory applications, since the performance bottleneck moves to the I/O performance. Virtualization adds a small overhead for MPI applications. However, the granularity of VMs, the number of virtual CPUs (vCPUs) per VM, seems to be important for MPI applications. Using multiple VMs on a physical system for the same instance of MPI applications may degrade the performance significantly for a subset of NPB applications.

2 Methodology

2.1 Target Multicore Architectures

We use two different types of commercial multicore systems to evaluate HPC applications with virtualization. The first system is a single-socket system with a 12-core AMD Opteron 6168 processor (`single-socket`). The processor is a multi-chip module with two dies packaged together. Each die has six cores. Each core has separate 64KB instruction and data caches, and 512KB L2 cache. Six cores in a die share a 6MB L3 cache. The twelve cores in the system have almost uniform memory latencies to any memory modules.

The second system (`dual-socket`) uses two Intel Nehalem E5530 processors, which have four cores in each processor. Each core has separate 32KB instruction and data caches, and a 256KB private L2 cache. Four cores in a processor share an 8MB L3 cache. In the dual-socket system, two quad-core processors are connected by QPI interconnections. With the QPI interconnections, each processor has its own DRAM memory banks. An important characteristic of the system is non-uniform memory access (NUMA). In such NUMA systems, the latencies of accessing remote memory connected to the other socket are much higher than the latency of accessing local memory.

2.2 Methodology

To evaluate the effects of virtualization, we use the Xen hypervisor (version 3.4.2) [6]. We compare the performance of two selected benchmarks on virtual-

ized configurations to that on non-virtualized (native) configurations. The guest operating system in the virtualized configurations is a Linux (kernel version 2.6.31.13) modified to support the para-virtualization mode of the Xen hypervisor. For the operating system in the native configurations, the same version of the Linux kernel is used.

We use two benchmarks representing different uses of HPC clusters: PARSEC [4] is a shared-memory multi-threaded benchmark. This benchmark is evaluated with a single physical machine. We use the largest input set for PARSEC (native input set). As a MPI-based benchmark, we evaluate the NAS parallel benchmark (NPB) [3]. To evaluate the overheads of MPI communications, we connected two systems by a 1gigabit Ethernet switch, and used the MPICH 1.2 library [2]. For NPB, we use the class C input set.

2.3 Virtual Machine Scheduling

In the Xen hypervisor, the unit of scheduling is a virtual CPU (vCPU). Each VM may have multiple vCPUs, emulating a multiprocessor system. The Xen hypervisor uses a credit-based scheduler, which assigns credits for each vCPU periodically to guarantee fairness among vCPUs. Since vCPUs are scheduled independently, there is no guarantee that the vCPUs from a single VM are scheduled together. The Xen hypervisor maintains queues for each physical core, but vCPUs may migrate to all the physical cores freely unless they are pinned to specific cores. In the default setting, the scheduler will try to maximize the overall throughput by not wasting any CPU cycles, ignoring the cost of migrating vCPUs to different cores. Whenever a core becomes idle, it will attempt to steal active vCPUs waiting in the queues of other cores.

In the target dual-socket system, relocating a thread across the processor boundary may cause two effects: shared L3 cache and NUMA effects. When a thread migrates from a processor to the other processor, it can no longer access the cached data in the L3 cache in the old processor directly. In the new processor, necessary data must be fetched either from the old L3 cache or the memory. The other effect is non-uniform memory access latencies. Depending on which memory modules a thread mostly accesses, the processor where the thread is running may have a significant effect on the overall performance due to non-uniform memory access latencies. In the target single-socket system, relocating a thread from a die to the other die causes only the effect of shared L3 cache, as the system has uniform memory latencies from all the cores.

3 Shared Memory Applications: PARSEC

3.1 Performance

Single Socket Results: To isolate the effect of NUMA, we first evaluate the effect of virtualization by using a system with one processor (single-socket). In the single-socket system, the memory access latencies from 12 cores are uniform

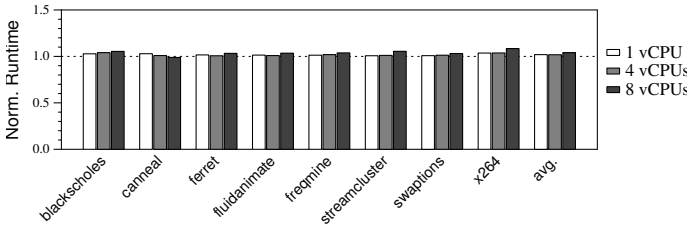

Fig. 1. Single socket (*unpinned* vCPUs): execution times with 1, 4, and 8 vCPUs

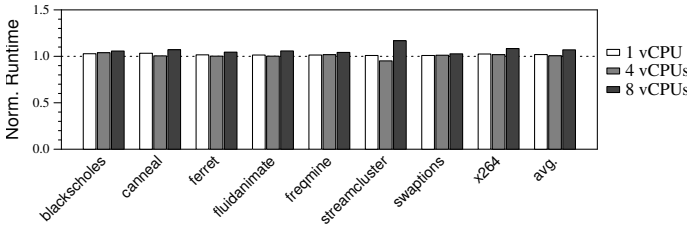

Fig. 2. Single socket (*pinned* vCPUs): execution times with 1, 4, and 8 vCPUs

regardless of memory banks. Among 12 cores, we use only 8 cores to be consistent with dual-socket results. Figure 1 presents the execution times of the PARSEC benchmark normalized to those of the native system with the same number threads. In this experiment, the vCPUs are not pinned to physical cores, and thus the Xen scheduler can migrate vCPUs without any restriction to minimize unused CPU cycles. For each application, three bars are shown: one, four, and eight vCPUs. The number of threads in each application is set to the number of vCPUs.

In general, for the single-socket system, the performance overheads by virtualization are insignificant, regardless of the number of vCPUs. For one and four vCPUs, the execution times increase by 2% on average compared to the native system, and for eight vCPUs, the average execution time increases by 4%. The Xen hypervisor supports efficient virtualization for compute-intensive shared-memory applications for the single-socket system with uniform memory access. To further investigate the effect of scheduling, we fix vCPUs to physical cores. Figure 2 presents the execution times normalized to those of the native system, when vCPUs are pinned to physical cores. The results are similar to those with the unpinned configuration. With uniform memory access latencies, mapping between vCPUs and physical cores does not have a significant impact on the performance of the PARSEC applications. Furthermore, the cost of vCPU migration across shared L3 caches is minor, as shown by the almost same performance by the pinned and unpinned configurations. In the unpinned configuration, vCPUs may migrate to the other die.

Dual Socket Results: To include the NUMA effect, we use a dual-socket system in which each socket has four cores. Figure 3 presents the execution times with the dual-socket system normalized to those of the native system. In this experiment, the vCPUs are not pinned to physical cores (and thus the

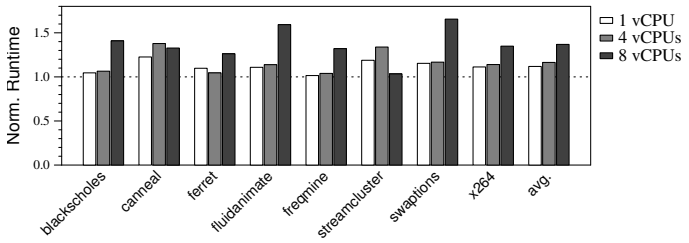

Fig. 3. Dual socket (*unpinned* vCPUs): execution times with 1, 4, and 8 vCPUs

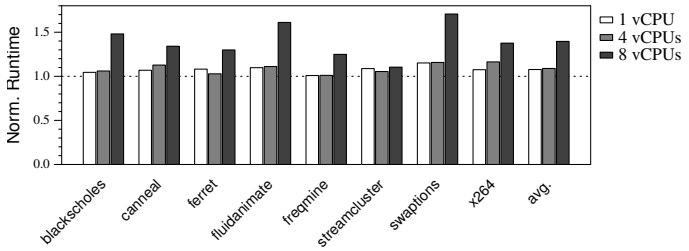

Fig. 4. Dual socket (*pinned* vCPUs): execution times with 1, 4, and 8 vCPUs

Xen scheduler can migrate the vCPUs as necessary). Unlike the previous single socket results, the performance overheads by virtualization are significant. Only when one vCPU is used, the performance degradation is relatively small 12% on average. However, when four and eight vCPUs are used, the performance degradations become 16% and 37% on average respectively.

To eliminate the effect of vCPU migration, we fix each vCPU to a physical core. In this case, the Xen scheduler cannot migrate vCPUs. Figure 4 presents the normalized execution times (to those of the native system) with the pinned configuration. For the one and four vCPU configurations, the performance degradations reduce to 8% and 9% respectively. However, for the eight vCPU configuration, the performance degradation increases slightly to 40%.

When the number of vCPUs is in the range from 1 to 4, pinning makes the system use only one socket for the vCPUs and allocate the most of the memory pages in the memory modules connected to the same socket. Therefore, pinning for 1-4 vCPUs reduces the effect of NUMA and eliminates the cost of vCPU migration across the shared L3 cache boundary. However, for 8 vCPUs, pinning may eliminate the cost of vCPU migration across the shared L3 cache boundary, but it does not mitigate the effect of NUMA. Eight vCPUs must use all the cores in both sockets, but the memory pages of the VM are mostly located in one of the socket. As shown in the single-socket results, the effect of vCPU migration across the L3 caches is minor compared to the effect of NUMA.

3.2 Mitigating the NUMA effect

In this section, we isolate the effect of NUMA to further investigate its performance impact on HPC applications. To explain the benefit of pinning in the

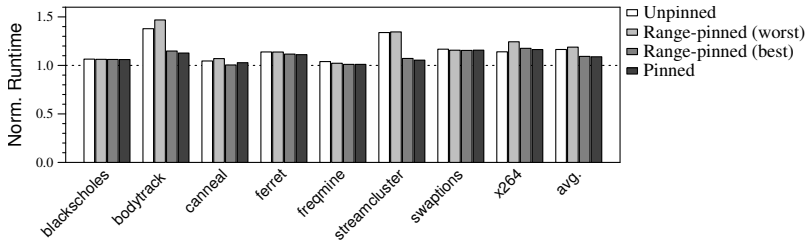

Fig. 5. The worst and best range pinning schemes for 4 vCPUs (dual-socket)

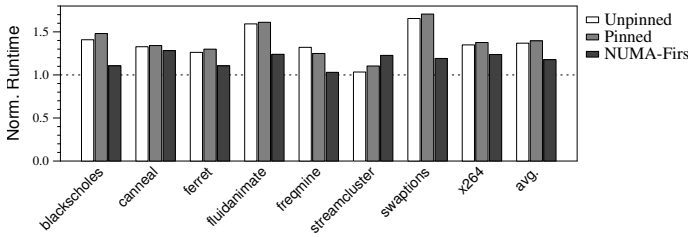

Fig. 6. *NUMA-first* optimization for 8 vCPUs (dual-socket)

dual-socket system (as shown in Figure 4), we evaluate the "worst" and "best" case scheduling for the four vCPU configuration. Considering the NUMA effect, the worst case scheduling is to map all four vCPUs on a socket, while all memory pages used by the vCPUs are located in the other socket. The best case scheduling is to map all four vCPUs on the same socket to which all the memory pages are located. Figure 5 presents the execution times normalized to those of the native system with the worst and best case scheduling for four vCPUs, as well as the unpinned and pinned configurations. For the worst and best scheduling, a vCPU is not fixed to a physical core, but fixed to a socket (range-pinned). In this case, a vCPU can be mapped to one of the four cores in the socket.

As shown in Figure 5, the performance with the unpinned configuration is slightly better than that with the worst case range-pinned configuration. However, the performance gap between the two configurations is very small. In the unpinned configuration, the default Xen hypervisor schedules vCPUs to any sockets, without considering the NUMA effect. The performance with the pinned configuration is similar to that with the best case range-pinned configuration. Pinning vCPUs has a similar effect to the best case configuration, since in our experiments, all four vCPUs happen to be mapped to the same socket to which their memory pages are located. However, we expect that blindly pinning vCPUs, without considering the memory affinity, will not improve performance consistently.

However, for the eight vCPU configuration, it is not possible to find the best case scheduling, since eight vCPUs must be mapped to 8 cores in two sockets. To reduce the effect of NUMA, we modified the Xen scheduler slightly such that it attempts to schedule vCPUs to the right socket. In the PARSEC applications, all the eight vCPUs are not always used, since available parallelism dynamically

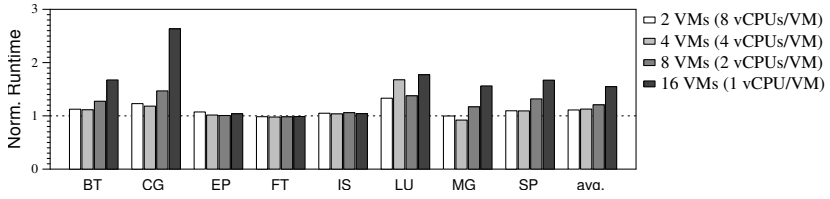

Fig. 7. NPB execution times (*unpinned* vCPUs): varying vCPUs per VM

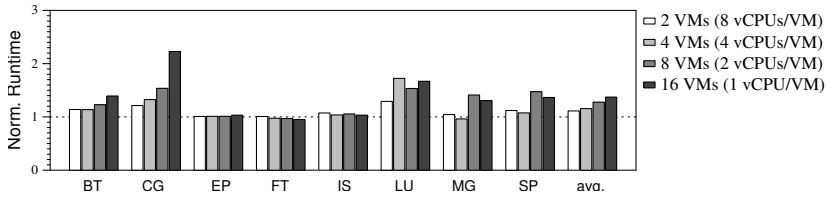

Fig. 8. NPB execution times (*pinned vCPUs*): varying vCPUs per VM

changes. If less than eight vCPUs are used, active vCPUs are scheduled as much as possible to the socket in which their memory pages reside. However, we do not make any physical core idle, if there are active vCPUs not scheduled to any core. Thus, if no core in the right socket is available, a vCPU will be scheduled to the other socket. This rudimentary optimization, called `NUMA-first`, provides a significant improvement in performance. Figure 6 presents the normalized execution times with the unpinned, pinned, and NUMA-first configurations. With the NUMA-first scheduling, the average performance degradation is reduced to 18% from 37% of the unpinned configuration. The NUMA-aware scheduling requires further investigation to make it adaptable to more complex cases than our configurations.

4 MPI-based Applications: NPB

In this section, we evaluate the performance overheads of virtualization with the MPI-based NPB. For these experiments, we use two physical systems connected by a 1gigabit Ethernet switch. For the experiments in this section, the half of the total MPI processes are running in a system, and the other half are running in the other system. For example, if the total number of MPI processes is 16, 8 processes are running in each physical system. For this evaluation of MPI applications, we use only dual-socket systems.

Unlike shared memory applications, which must run on a single virtual machine, MPI-based applications can run with various numbers of virtual machines per system. With two dual-socket systems, up-to 16 MPI processes run without sharing cores. For 16 MPI processes, in each system, 8 MPI processes can use a VM with 8 vCPUs, two VMs with 4 vCPUs, four VMs with 2 vCPUs, or eight VMs with 1 vCPU. To support the same number of MPI processes, different granularities of VMs can be used. Such VM granularity, or the number of

vCPUs per VM, may have some impact on the cost of communication among MPI processes. MPI communications among processes in a VM are done only within the guest operating system. MPI communications among the VMs in a system do not access the network hardware, but the communications must pass through the hypervisor and the domain0 VM. MPI communications among the VMs in different systems must access the network hardware, hypervisor, and the domain0 VM.

Figure 7 presents the execution times with different numbers of VMs for 16 MPI processes. For this result, vCPUs are not pinned to any cores. Firstly, for all applications, using the largest VM (8 vCPUs per VM) is better than using multiple VMs in a system. It is because MPI communications within a VM have lower overheads than those across VMs. Secondly, for each application, if the best VM granularity (8 vCPUs per VM) is used, the performance overheads on MPI-applications by virtualization are much lower than those on shared memory applications. Even though all the 8 cores are used for each system, the average execution time is only 11% higher than that of the native system. Although these MPI applications also suffer from the effect of NUMA, its effect on the overall performance is relatively low, since the performance is also dependent upon the performance of MPI communications. Due to the I/O activities, NUMA effect does not dominate the overall performance.

Figure 8 presents the execution times with vCPUs pinned to physical cores. Pinning vCPUs does not improve the NPB performance for the best VM granularity (8 vCPU per VM), with a similar 11% average increase of execution times. However, pinning improves performance for any VM granularity other than 8 vCPU per VM.

5 Related Work

The effects of virtualization on the performance of applications have been studied in previous work. Due to space limitation, we review some of such work in this section. Huang et al showed that I/O virtualization overhead is the major issue for virtualization by evaluating the performance of NPB, and proposed VMM-bypass I/O to reduce I/O virtualization overhead [9]. In [12], the effects of resource sharing on the performance of HPC applications were studied in a virtualized multicore cluster. Specially, the authors focused on how the communications of HPC applications are affected when multiple VMs share a single Infiniband interconnect. In [14], the performance of the compute-bound benchmark applications was analyzed, and in [11], the performance overheads for network I/O device virtualization were measured. A simulation-driven approach was presented in [7], which analyzes the virtualization overheads of I/O intensive workloads. The performance impact of a consolidated workload, which is composed of server applications such as a web server and a database server, was evaluated in [5]. In our paper, we focus on how the complex memory hierarchy affects the performance of HPC applications in virtualized systems, which is not considered in the above previous work.

A VM-aware MPI library was developed in [8]. It can reduce the communication overhead for HPC applications, by allowing VMs in the same physical host to communicate via shared memory. To improve I/O performance, Liao et al presented cache-aware scheduling which co-schedules Dom0 and I/O intensive DomUs to communicate more efficiently via a last level cache, and credit-stealing which steals credits for I/O intensive vCPUs [10]. It is our future work to develop VM scheduling techniques that are aware of memory hierarchy in multicore systems for HPC applications.

The performance of a cloud computing service, Amazon EC2, has been evaluated in [15, 16]. The results show the performance degradation of NPB (in [15]), and unstable network throughput (in [16]), indicating that the cloud computing service is not mature yet.

In [13], the implication of varying the number of application processes per VM was studied, but unlike our work, only two simple configurations, two VMs with 1 process each and one VM with two processes, were used.

6 Conclusion

In this paper, we evaluate single and dual socket multicore systems with the Xen hypervisor. For shared memory applications in dual-socket systems, NUMA awareness is critical for performance, while the overheads of virtualization are minor in single socket systems. As the complexity and non-uniformity in memory systems increase, the importance of NUMA and cache awareness in VM scheduling will become critical, especially for shared memory applications. For MPI-based applications, the NUMA effect is much smaller than that with the shared memory applications. However, the granularity of VMs (the number of vCPU per VM) becomes critical for the overall performance

References

1. Amazon EC2. http://aws.amazon.com/ec2/.
2. MPICH. http://www.mcs.anl.gov/research/projects/mpich2/.
3. The NAS Parallel Benchmarks. http://www.nas.nasa.gov/Resources/Software/npb.html/.
4. The Princeton Application Repository for Shared-Memory Computers (PARSEC). http://parsec.cs.princeton.edu/.
5. P. Apparao, R. Iyer, X. Zhang, D. Newell, and T. Adelmeyer. Characterization & analysis of a server consolidation benchmark. In *VEE '08: Proceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, pages 21–30, New York, NY, USA, 2008. ACM.
6. P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 164–177, New York, NY, USA, 2003. ACM.
7. V. Chadha, R. Illiikkal, R. Iyer, J. Moses, D. Newell, and R. J. Figueiredo. I/o processing in a virtualized platform: a simulation-driven approach. In *VEE '07: Proceedings of the 3rd international conference on Virtual execution environments*, pages 116–125, New York, NY, USA, 2007. ACM.

8. W. Huang, M. J. Koop, Q. Gao, and D. K. Panda. Virtual machine aware communication libraries for high performance computing. In *SC '07: Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, pages 1–12, New York, NY, USA, 2007. ACM.
9. W. Huang, J. Liu, B. Abali, and D. K. Panda. A case for high performance computing with virtual machines. In *ICS '06: Proceedings of the 20th annual international conference on Supercomputing*, pages 125–134, New York, NY, USA, 2006. ACM.
10. G. Liao, D. Guo, L. Bhuyan, and S. R. King. Software techniques to improve virtualized i/o performance on multi-core systems. In *ANCS '08: Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, pages 161–170, New York, NY, USA, 2008. ACM.
11. A. Menon, J. R. Santos, Y. Turner, G. J. Janakiraman, and W. Zwaenepoel. Diagnosing performance overheads in the xen virtual machine environment. In *VEE '05: Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments*, pages 13–23, New York, NY, USA, 2005. ACM.
12. A. Ranadive, M. Kesavan, A. Gavrilovska, and K. Schwan. Performance implications of virtualizing multicore cluster machines. In *HPCVirt '08: Proceedings of the 2nd workshop on System-level virtualization for high performance computing*, pages 1–8, New York, NY, USA, 2008. ACM.
13. A. Tikotekar, H. Ong, S. Alam, G. Vallée, T. Naughton, C. Engelmann, and S. L. Scott. Performance comparison of two virtual machine scenarios using an hpc application: a case study using molecular dynamics simulations. In *HPCVirt '09: Proceedings of the 3rd ACM Workshop on System-level Virtualization for High Performance Computing*, pages 33–40, New York, NY, USA, 2009. ACM.
14. A. Tikotekar, G. Vallée, T. Naughton, H. Ong, C. Engelmann, and S. L. Scott. An analysis of hpc benchmarks in virtual machine environments. pages 63–71, 2009.
15. E. Walker. Benchmarking Amazon EC2 for HP Scientific Computing. *;LOGIN: The USENIX Magazine*, 33(5):18–23, 2008.
16. G. Wang and T. S. E. Ng. The impact of virtualization on network performance of amazon ec2 data center. In *INFOCOM'10: Proceedings of the 29th conference on Information communications*, pages 1163–1171, Piscataway, NJ, USA, 2010. IEEE Press.