Virtualizing HPC applications using modern hypervisors

Alexander Kudryavtsev, Vladimir Koshelev, Boris Pavlovic and Arutyun Avetisyan
Institute for System Programming, Russian Academy of Sciences
109004, Moscow, Alexander Solzhenitsyn st., 25
Russian Federation
{alexk,vedun,bpavlovich,arut}@ispras.ru *

ABSTRACT

In this paper we explore the prospects of virtualization technologies being applied to high performance computing tasks. We use an extensive set of HPC benchmarks to evaluate virtualization overhead, including HPC Challenge, NAS Parallel Benchmarks and SPEC MPI2007. We assess KVM and Palacios hypervisors and, with proper tuning of hypervisor, we reduce the performance degradation from 10-60% to 1-5% in many cases with processor cores count up to 240. At the same time, a few tests provide overhead ranging from 20% to 45% even with our enhancements.

We describe the techniques necessary to achieve sufficient performance. These include host OS tuning to decrease noise level, using nested paging with large pages for efficient guest memory allocation, and proper NUMA architecture emulation when running virtual machines on NUMA hosts.

Comparing KVM/QEMU and Palacios hypervisors, we conclude that in general the results with proper tuning are similar, with KVM providing more stable and predictable results while Palacios being much better on fine-grained tests at a large scale, but showing abnormal performance degradation on a few tests.

Categories and Subject Descriptors: D.4.7 [Operating Systems]: Organization and Design

General Terms: Design, Experimentation, Measurement, Performance.

Keywords: Virtual machine monitors, high performance computing, parallel computing.

1. INTRODUCTION

Virtualization technologies are getting more mature and their capabilities have grown rapidly in the last few years. Virtualization gets new applications in different areas. Hardware-assisted virtualization technologies are getting capable of providing near-native performance for some classes of applications. As a result, researchers started to investigate the capabilities and limitations of virtualization when applied to High Performance Computing (HPC) tasks.

Benefits of applying virtualization into HPC area are being widely discussed [17, 7]. Fault tolerance, compatibility and flexibility are among them. Another idea is to employ cloud concept for building HPC clouds which provide scalability, cost effectiveness and ease of access. Recent research shows that HPC virtualization is feasible for at least some classes of applications [17].

Currently there is a substantial lack of experimental data in HPC virtualization area. Different applications have to be tested on different hardware using a variety of hypervisors to fully understand the limitations of existing virtualization technologies. Modern multi-socket hardware provide additional requirements to virtualization software, including Non-Uniform Memory Access (NUMA) emulation in guest system. Proper NUMA emulation is really important for VM performance when running on multi-socket NUMA hardware because of different memory access latencies for CPU accessing its local memory and other CPUs' memory.

In this work we explore the performance of HPC applications running inside a set of VMs in a cluster. Our primary goal is to evaluate full-system virtualization overhead, to understand the reasons of this overhead, and to minimize it if possible. We assess Kernel Virtual Machine (KVM) [9] and Palacios hypervisor [11], which was developed specially for HPC systems. It should be noted that we used a modified version of Palacios since the original version does not yet supports many features required to launch it on our test hardware. The changes we made to Palacios are briefly described in the following sections.

To achieve maximum performance compared to the native case, we allocate virtual machines (VMs) as much resources as possible, including all processor cores, and pass high-speed interconnect device into VM using Intel VT-d in KVM and paravirtualization in Palacios. Also we expose an emulated NUMA architecture into the VM and set it up to reflect the real NUMA configuration of our test hardware.

We use HPC Challenge (HPCC) benchmark [12], NAS Parallel Benchmarks (NPB) [4] and SPEC MPI2007 [13] benchmark as a test suite since these benchmarks are widely used as HPC system performance indicators. Performance tests were made on HPC clusters containing 8 and 20 Intel Xeon nodes (up to 240 processor cores used) connected to a 40 Gb/s Infiniband network. Many previous results were gathered on single node systems, which does not fully evaluate virtualization overhead.

*The work was supported by The Ministry of education and science of Russia under the contract No. 07.524.11.4018.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Workshop on Cloud Services, Federation, and the 8th Open Cirrus Summit, September 21, 2012, San Jose, CA, USA.
Copyright 2012 ACM 978-1-4503-1267-7 ...$10.00.

Our main contributions are the following:

- We describe the techniques necessary to achieve sufficient performance using KVM/QEMU. These include host OS tuning to decrease noise level, using nested paging with large pages for efficient guest memory allocation, and proper NUMA architecture emulation when running virtual machines on NUMA hosts.
- We provide virtualization overhead test results for an extensive set of HPC applications.
- We compare the performance of Kitten HPC OS / Palacios hypervisor and Linux / KVM/QEMU hypervisor. We show that KVM/QEMU provides more stable and predictable results, while Palacios is much better on fine-grained tests, especially at large scale.
- We examine gathered test data and investigate the reasons of observed overhead caused by virtualization.

The remainder of this document is organized as follows. In the next section, the related work is discussed. Section 3 describes hypervisor systems which we use and some basic methods for performance optimization. Section 4 describes performance evaluation methodology and provides test results together with discussion. Section 5 contains conclusions.

2. RELATED WORK

A thorough analysis of present work in the area of HPC virtualization research is done in the paper by Andrew J. Younge et al. [17]. The authors note that current performance test results available in articles are sometimes conflicting with each other, and try to perform unbiased assessment of modern virtualization technologies, including Xen [5], KVM, VirtualBox [16]. The authors used HPCC and SPEC OpenMP benchmark suites, performance loss for High Performance Linpack (HPL) is about 30% on 8-core system. It should be noted that the authors performed all tests on a single node system, which does not allow one to evaluate performance loss while scaling number of nodes.

The authors of [17] point out that virtualization is feasible for at least some HPC applications and choose KVM as well-suited hypervisor for such tasks. From our experience, KVM is becoming one of the most full-featured open source hypervisor. When compared to Xen, KVM is much easier to manage and understand since it is built into mainline Linux kernel.

Regola and Ducom evaluate the I/O performance of KVM, Xen, OpenVZ [1] and Amazon's EC2 "Cluster Compute Node" [15]. The authors note that CPU virtualization performance is already studied well while I/O virtualization should be studied more thoroughly and in the paper the performance of disk and network I/O is evaluated. Also additional overhead of Intel's VT-d when passing Infiniband Host Channel Adapter (HCA) into VM is evaluated for the first time. NAS Parallel Benchmark MPI results show that the worst overhead for KVM is 30 times compared to the native case, for Xen — 50%. It seems that the authors used some outdated version of KVM/QEMU because our results are more optimistic.

OpenVZ I/O performance with Infiniband was not tested since it does not have Infiniband driver, but the authors claim that it has near-native I/O performance in almost all cases. We agree with that, since container-based virtualization is much more lightweight than full-system virtualization, but we believe that hardware-assisted virtualization technologies will evolve together with software to provide near-native I/O performance.

For example, as Lange et al. note [11], the main reason for performance overhead of device passed inside VM is the interrupt overhead. On every interrupt, CPU has to transfer control to the hypervisor. When the guest finishes interrupt handling, the end of interrupt signal is also handled by the hypervisor. This chain may significantly delay interrupt delivery, and it increases the latency. Gordon et al. proved this assumption in their Exit-Less Interrupt (ELI) system [8], which allows to pass interrupts inside VM without exit to the hypervisor. The authors' results show that ELI can decrease performance overhead of KVM/QEMU from 40 to 1-2% when running one core VM with Netperf, Apache and Memcached applications used as a benchmark. This performance overhead is caused by a huge number of interrupts issued by passed inside VM Ethernet card.

Virtualization of a large scale supercomputer was studied by Lange et al. [11] for the first time using a specially crafted Palacios hypervisor together with Kitten HPC OS [10] used as a host OS. Palacios hypervisor is developed as a part of the V3VEE project [3]. The authors gathered test results for a list of HPC applications and benchmarks, and the virtualization overhead was less than 5% with node count up to 4096 nodes. However it should be noted that these results were gathered with only one virtual CPU per one 4-core node.

The authors of Palacios also note the importance of OS noise problem. The impact of OS noise on parallel applications is being widely studied [6, 14]. In general, the noise impact is highly dependent on application computation-communication ratio, communication granularity, process grid characteristics, and for some applications even minor noise can lead to significant performance and scalability degradation. Thus, when virtualizing an HPC system, the host OS noise problem should be taken into account. From this point of view, Kitten OS should perform better as a host OS than Linux since it was designed specially for HPC.

3. VIRTUALIZATION SYSTEMS UNDER CONSIDERATION

The most widespread virtualization systems, in general, provide almost the same functionality. Xen and KVM seem to be the most popular hypervisors (among open source solutions) which are heavily used in production. As it was noted before, we decided to investigate KVM as a hypervisor for HPC applications. In the future work we hope to evaluate Xen too. Also we studied the Palacios hypervisor, which is created for computer architecture research and use in high performance computing.

KVM/QEMU

KVM hypervisor is a full virtualization solution for Linux which supports hardware-assisted virtualization extensions (Intel VT-x and AMD-V). KVM consists of several loadable kernel modules and provides only hardware-virtualized CPUs. The rest of the VM is implemented by QEMU emulator. QEMU can run VMs using KVM hardware-assisted

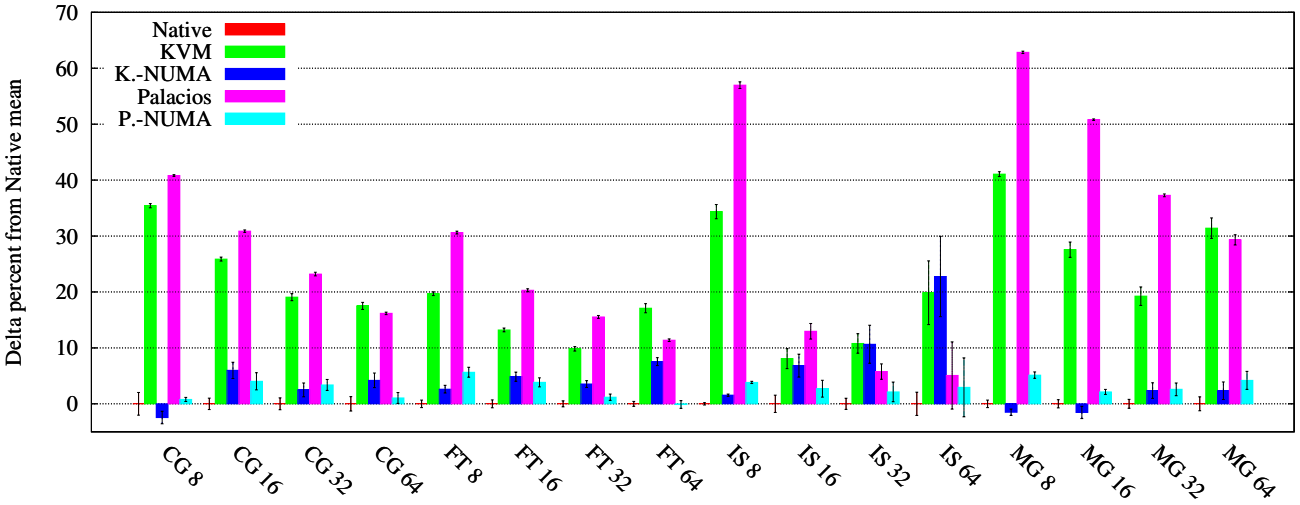

Figure 1: KVM/QEMU and Palacios results for NPB CG, FT, MG, IS tests.

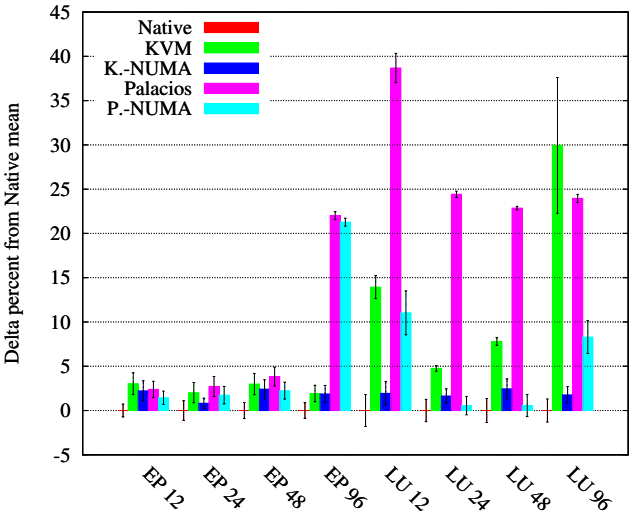

Figure 2: KVM/QEMU and Palacios results for NPB EP, LU tests.

virtualization or binary translation technique.

KVM/QEMU together with Linux allows real devices to be assigned to (or passed into) the VM. To support direct device assignment, the host hardware must contain IOMMU (I/O Memory Management Unit) — either Intel's VT-d or AMD's IOMMU. The main goal of the IOMMU is to map the device address space into guest physical address space using page tables. To allow guest system to use Infiniband HCA, we pass it into VM using Intel's VT-d.

The x86_64 virtual memory system has support for different page sizes. Linux kernel uses 4KB pages by default and contains feature called HugeTLB to provide larger pages (or *huge* pages) on demand. Huge pages may be used to decrease both the number of memory accesses required for guest physical address (PA) to host PA translation and the number of TLB misses when using nested paging.

The Linux kernel supports two ways of using HugeTLB: Transparent Hugepages and HugeTLBfs. The Transparent Hugepages mechanism allows the kernel to allocate anonymous memory using huge pages without the need to modify the application if the requested memory size is page size aligned. HugeTLBfs uses reserved huge pages via file system. Custom programs can map files from HugeTLBfs and these mappings will be backed by huge pages. The version of QEMU which we use (1.0.1) allocates memory for VM via the call to `posix_memalign` with 2MB alignment, thus Transparent Hugepages mechanism works by default.

Expose the real NUMA topology to VM.

A virtual SMP system running on top of a real NUMA system may suffer from serious performance degradation. Each NUMA node has its own CPU set and memory ranges and access from one node's CPU to other node's memory requires more time than access to the same node's memory. QEMU has support for NUMA system emulation, but emulated structure does not correspond to the real hardware topology. To solve this problem we pin virtual CPUs to distinct physical cores to disallow QEMU's threads migration between cores and employ `mbind` system call to be sure that selected memory ranges of VM memory will be allocated at corresponding nodes. We found that `mbind` ruins Transparent Hugepages allocation, so we allocate memory directly from HugeTLBfs using QEMU's `-mempath` parameter.

Palacios

The Palacios hypervisor [11] was developed with the goal to effectively virtualize HPC applications. The distinguishing feature of Palacios is its embeddable nature achieved via a set of unified host OS interfaces. Initially Palacios was embedded into Kitten HPC OS, but in the latest release Linux support is implemented too. We decided to assess Palacios since we believe it is a promising direction. We use Kitten OS as a host OS since this system should generate much less noise when compared to Linux – this advantage is the most interesting for our experiments.

Palacios supports nested paging and a number of shadow paging techniques. For our tests, we used nested paging with 2MB pages since it is the best choice for the Linux guest [11].

Device assignment technique.

The real communication device should be passed into VM to achieve performance which is comparable to the native

run. To make Palacios device assignment possible, special paravirtual interface is used. The memory for guest system is allocated in one physically contiguous range. The guest system can use hypercall to get its physical memory offset in the real physical memory. Using this offset, guest OS can patch addresses for DMA transactions of selected devices. In fact, changes required in guest OS to implement this interface are relatively simple. We created our own patch for the Linux kernel.

Since Kitten OS does not provide drivers for the most hardware, we had to pass some devices among Infiniband HCA inside the VM, namely SATA hard disk controller and Ethernet card.

Problems with Palacios.

As soon as we started our experiments with Palacios release 1.2, we faced some difficulties, including incomplete device assignment support, incomplete VT-x support, maximum $\sim$ 3.5GB of RAM, no NUMA support. We created our local branch and implemented these features which were required to launch VM on our test hardware[1].

One serious problem that was not solved at all is the problem with timing in the guest system due to the poor timer implementation in Palacios. Currently the guest clock is inaccurate, in our case clock skew was about 30-40 minutes per one day. To check the time skew when running tests, we used NTP time synchronization and checked that the time offset is not too large.

4. PERFORMANCE EVALUATION AND DISCUSSION

We tried to cover a wide range of HPC applications using HPCC, NPB and SPEC MPI2007 benchmarks in different configurations. The following subsections describe our experimental setup, testing methodology and results. At the end of this section, we analyze and discuss the reasons for performance overhead in different cases.

Experimental setup

We use two HP clusters as a testbed. First cluster consists of 8 HP ProLiant BL2x220c G7 blades. For tests we used up to 8 nodes. Each node contains 2 Intel Xeon X5670 CPUs (6 cores per CPU) and 24GB of RAM. Hyperthreading was disabled on all nodes. For running SPEC MPI2007 we used second cluster which consists of 20 HP ProLiant SL390s G7 nodes, each containing 2 Intel Xeon X5650 CPUs (6 cores per CPU) and 24GB of RAM. Cluster nodes communicate via 1Gbit/s. service Ethernet network, and 40Gbit/s. Infiniband network used for computing.

The systems used for native tests, as a host OS for KVM and as a guest OS are Linux CentOS 6.0-6.3. The kernel is modified to support Palacios device assignment. MPI library is OpenMPI 1.4.3, GCC version used to build test suites is 4.4.4. Infiniband stack is provided by Mellanox OFED for Linux, version `1.5.3-1.0.0-rhel6-x86_64`. QEMU version is `qemu-kvm-1.0.1` with our modifications. The Kitten OS version is 1.2.0 with modifications required to support device assignment in Palacios' guests. The guest system is configured with 16GB of RAM, 12 processor cores, with or without NUMA architecture emulation.

[1]We contributed our code to the V3VEE project, some of our patches could be found at the V3VEE project Web site [3] and some are already integrated into the development branch.

Benchmarks.

To measure the performance we use HPC Challenge, NAS Parallel Benchmark suites (first cluster, running on 2, 4 and 8 nodes) and SPEC MPI2007 suite (second cluster, running on 4, 8, 16, 20 nodes).

NPB version is 3.3.1 for MPI (NPB3.3-MPI). From NPB suite we employ IS, EP, CG, MG, FT and LU tests with the problem class C. IS stands for Integer Sorting. EP is Embarrassingly Parallel Gaussian random variates generator. CG is a program using Conjugate Gradient method. MG (MultiGrid) approximates the solution to discrete Poisson equation. FT is discrete 3D fast Fourier Transform with all-to-all communication. LU is a solver for systems of partial differential equations. We run 8, 9 or 12 processes per node depending on test constraints. Results are expressed in seconds.

HPCC version is 1.4.1 with ATLAS 3.8.4. Tests from HPCC suite are executed using 12 processes per node. We assess the results of STREAM, RandomAccess and HPL tests from HPCC package. STREAM measures sustainable memory bandwidth. We use data from Single (single process is computing) and EP (Embarrassingly Parallel — all processes compute the same thing independently) versions of this test when running on one node. RandomAccess measures the rate of integer updates to random memory areas in GigaUpdates per second (GUP/s). This test is known to be very challenging for virtualization since it provides huge TLB pressure. HPL measures the rate of solving for linear system of equations. Results are provided in GFlops. For HPL we use the following parameters: matrix size is 30000, block size is 150, computation grid is square when it is possible. We exclude other HPCC benchmarks' results due to the space limitations.

SPEC MPI2007 version is 2.0. Currently we gathered data on native system and KVM system with NUMA emulation. We selected GemsFDTD, lammps, pop2, socorro, tachyon, wrf2, zeusmp2 medium size tests from SPEC MPI2007 package. Tests description is available at SPEC website [2].

Virtual machines are known to have problems with timing. Hypervisor and host OS provide additional noise source and cause guest timers to be less precise. As a result, run times in virtual environment tend to be more scattered. To make our results more precise, we perform most of NPB runs 50 times, HPCC runs 20 times and SPEC runs 10 times. We report the estimate of the mean together with 97% confidence interval for the mean value calculated using Student's t distribution to understand the reliability of gathered data. Though consequent test runs cannot be independent, confidence interval could help us to estimate the influence of different measurement error sources and to be sure that we did enough consequent runs.

Results

We started our testing with the "default" QEMU configuration, using default memory allocation mechanism. Also initially we tried virtual CPU pinning to prevent QEMU threads migration, and checked that HugeTLBfs mechanism provides the same overhead as Transparent Hugepages. Results for all these configurations were almost the same, with up to 50% performance overhead in some cases. After that

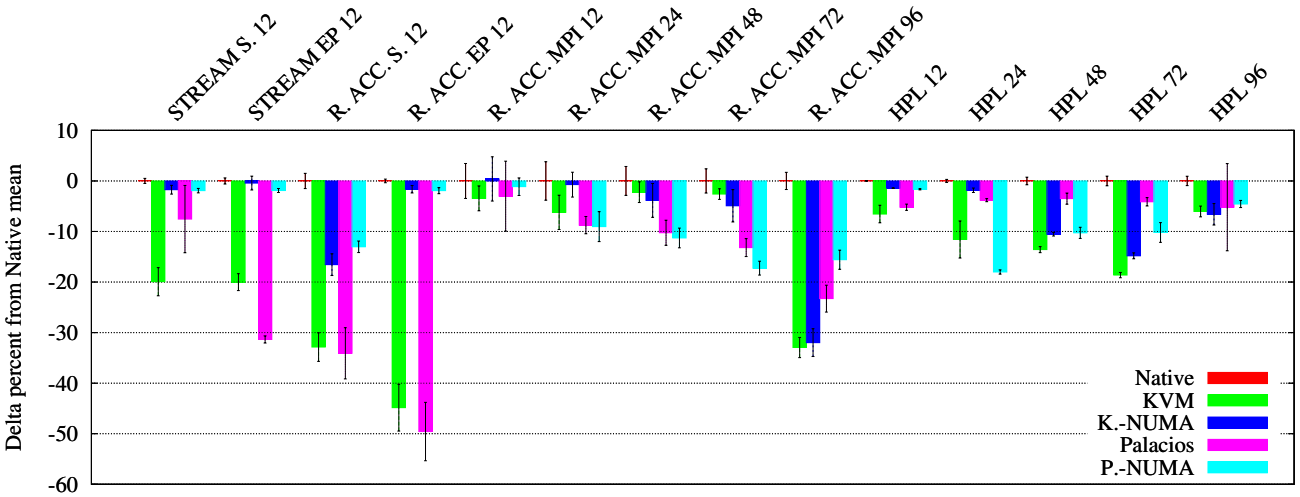

Figure 3: KVM/QEMU and Palacios results for HPCC suite.

we figured out that the reason for most overhead is the NUMA architecture of our test hardware. QEMU emulates an SMP system by default, which causes performance penalties when some virtual CPU accesses non-local memory. The next step was to provide the NUMA architecture to virtual machines corresponding to the real NUMA configuration. It was performed using QEMU NUMA emulation support, as described in the previous section.

For Kitten/Palacios system we tested two configurations – with and without NUMA support. In both cases 2MB nested pages were used for the guest memory. Overall NPB results comparing configurations of KVM/QEMU and Palacios are provided in figures 1, 2 with error bars denoting 97% confidence interval for each test mean. Data is plotted in relation to the Native case which has the value of zero in each bar group. Key under each bar group consists of the test name and process count used for computation. Results for HPCC suite are in Figure 3. "R. ACC." stands for Random Access, "S." is a Single version of the test.

The first conclusion for NPB suite results is that NUMA-awareness matters. Without NUMA Palacios and KVM show more than 60% and 40% overheads on MG 8 test, but with NUMA this overhead disappears. The same situation is true for the most other tests. Another fact concerning NUMA is that for almost all tests without NUMA, overhead decreases while the number of processes grows (the difference for one test could be 30% and more, see IS and MG tests). At the same time, for NUMA case in many tests overhead varies slightly.

For HPCC tests, NUMA emulation improves performance results for STREAM and Random Access Single and EP versions. For Random Access EP, NUMA-enabled guest almost eliminates 45-50% overhead of the non-NUMA guest. For MPI Random Access, the situation is not so definite – NUMA-enabled guest may perform better or worse.

The next thing we can conclude is that Palacios implementation has some drawbacks when compared to KVM – it can be seen when comparing results without NUMA support for almost all NPB tests and HPCC Random Access test with 24, 48 and 72 processes. The maximum difference is about 24% on LU 12 test. Also Palacios shows strange results for EP 96 test. HPL results are strange too: on 24, 48 and 72 processes Palacios without NUMA emulation shows the best result, but with NUMA emulation enabled overhead increases a few times. Our hypothesis is that KVM/QEMU has an important advantage: Linux host kernel automatically arranges memory and CPUs in the "default" QEMU configuration, while Palacios's VM has predefined virtual CPU to physical core mapping and memory ranges.

We can also see that sometimes the results inside VM with NUMA support are better than native. For example, look at tests CG, MG, MG, where KVM behaves better than native. Probably the reason for this behavior is that some unaccounted factors were present during these runs.

Finally, the CG, FT and IS test results with 64 processes and Random Access MPI results with 96 processes show the advantage of Kitten with Palacios over KVM/QEMU in the NUMA case. This advantage ranges from 3.5 to 21%. Probably this advantage is associated with the decreased noise of Kitten OS when compared to Linux, especially for fine-grained IS test. In general, we can conclude that KVM provides more stable and predictable results, while Palacios is better on fine-grained tests. Meanwhile, Palacios shows abnormal performance degradation on some tests.

SPEC MPI2007 results.

Our current SPEC MPI2007 results are available only for KVM with NUMA emulation enabled with up to 240 cores used for computation. The results are presented in Figure 4. We can see that GemsFDTD, lammps, tachyon and wrf2 scale well with persistent overhead around 5-10%. At the same time, other three tests show noticeable overhead ranging from 20 to 43%. In fact, we see the clear division of applications into two classes; among these seven tests, more than a half can be virtualized successfully. For the rest tests, the reasons for overhead should be thoroughly investigated.

5. CONCLUSION

Our primary contribution has been to demonstrate the importance of proper hypervisor and host OS tuning when running HPC task inside virtual machines, including NUMA architecture emulation according to the real configuration. In particular, we explored KVM/QEMU and Palacios hypervisors. KVM is widely used in industry while Palacios is

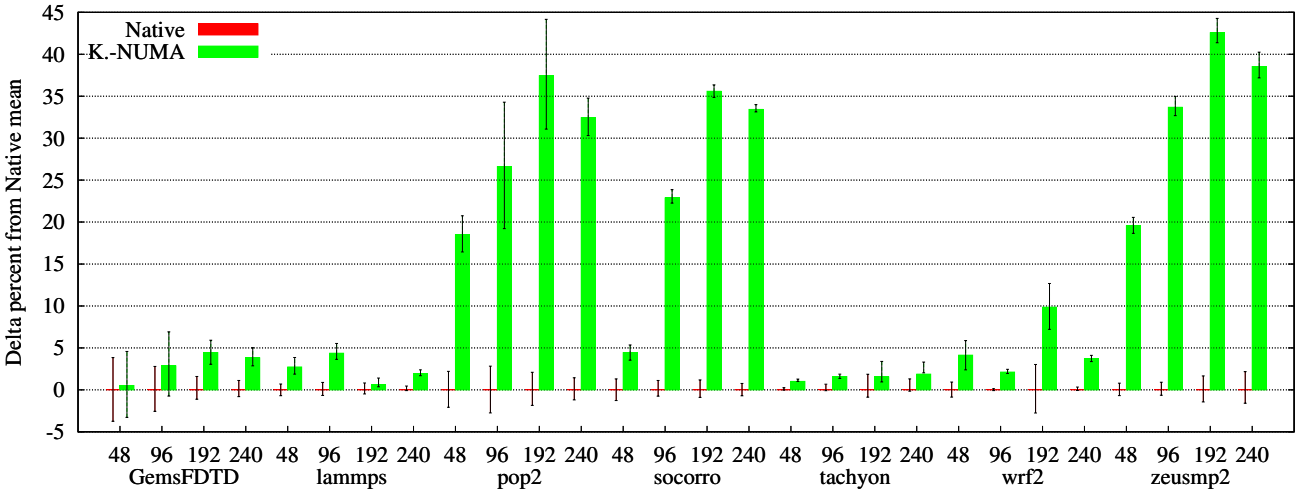

Figure 4: KVM/QEMU results for SPEC MPI2007 suite.

a young project targeted at HPC virtualization. We patched QEMU and Palacios to make the real NUMA topology available in the guest system. By using proper NUMA emulation, we reduced the performance degradation from 10-60% to 1-5% on many tests from HPCC and NPB suites. At the same time, SPEC MPI2007 results show that some applications still suffer from overhead ranging from 20% to 45%, while other scale well with core count up to 240.

We assess KVM/QEMU and Palacios hypervisors and conclude that in general their results with NUMA emulation enabled are similar, with KVM providing more stable and predictable results and Palacios being much better on fine-grained tests, but showing abnormal performance degradation on some other tests. The noise of the host OS is really important for fine-grained tests scalability and in this respect Kitten behaves better than Linux resulting in better scaling for tests running inside Palacios' virtual machines. We believe that the noise amount generated by virtualization system will become crucial for successfull virtualization of large-scale HPC systems.

6. REFERENCES

[1] OpenVZ: container-based virtualization for Linux, http://openvz.org/.

[2] SPEC MPI2007 Documentation, http://www.spec.org/mpi/Docs/.

[3] V3VEE: An Open Source Virtual Machine Monitor Framework For Modern Architectures, http://v3vee.org/.

[4] D. Bailey, T. Harris, W. Saphir, R. van der Wijngaart, A. Woo, and M. Yarrow. The NAS parallel benchmarks 2.0. Technical Report NAS-95-020, NASA Ames Research Center. December 1995.

[5] 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. *SIGOPS Oper. Syst. Rev.*, 37:164–177, Oct. 2003.

[6] K. Ferreira, P. Bridges, and R. Brightwell. Characterizing application sensitivity to OS interference using kernel-level noise injection. In *International Conference for High Performance Computing, Networking, Storage and Analysis, 2008.*, pages 1 –12, November 2008.

[7] A. Gavrilovska, S. Kumar, H. Raj, K. Schwan, V. Gupta, R. Nathuji, R. Niranjan, A. Ranadive, and P. Saraiya. Abstract High-Performance Hypervisor Architectures: Virtualization in HPC Systems. In *1st Workshop on System-level Virtualization for High Performance Computing (HPCVirt), in conjunction with EuroSys 2007*, 2007.

[8] A. Gordon, N. Amit, N. Har'El, M. Ben-Yehuda, A. Landau, A. Schuster, and D. Tsafrir. ELI: Bare-Metal Performance for I/O Virtualization. In *Proceedings of the Seventeenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2012)*, 2012 (to appear).

[9] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori. KVM: the Linux virtual machine monitor. In *OLS '07: The 2007 Ottawa Linux Symposium*, pages 225–230, July 2007.

[10] J. Lange, K. Pedretti, T. Hudson, P. Dinda, Z. Cui, L. Xia, P. Bridges, A. Gocke, S. Jaconette, M. Levenhagen, and R. Brightwell. Palacios and Kitten: New high performance operating systems for scalable virtualized and native supercomputing. In *2010 IEEE International Symposium on Parallel Distributed Processing (IPDPS)*, pages 1 –12, April 2010.

[11] J. R. Lange, K. Pedretti, P. Dinda, P. G. Bridges, C. Bae, P. Soltero, and A. Merritt. Minimal-overhead virtualization of a large scale supercomputer. In *Proceedings of the 7th ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, VEE '11, pages 169–180, New York, NY, USA, 2011. ACM.

[12] P. R. Luszczek, D. H. Bailey, J. J. Dongarra, J. Kepner, R. F. Lucas, R. Rabenseifner, and D. Takahashi. The HPC Challenge (HPCC) benchmark suite. In *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, SC '06, New York, NY, USA, 2006. ACM.

[13] M. S. Müller, M. van Waveren, R. Lieberman, B. Whitney, H. Saito, K. Kumaran, J. Baron, W. C. Brantley, C. Parrott, T. Elken, H. Feng, and C. Ponder. SPEC MPI2007 – an application benchmark suite for parallel systems using MPI. *Concurr. Comput. : Pract. Exper.*, 22(2):191–205, Feb. 2010.

[14] F. Petrini, D. Kerbyson, and S. Pakin. The Case of the Missing Supercomputer Performance: Achieving Optimal Performance on the 8,192 Processors of ASCI Q. In *2003 ACM/IEEE Conference on Supercomputing*, page 55, November 2003.

[15] N. Regola and J.-C. Ducom. Recommendations for virtualization technologies in high performance computing. In *Proceedings of the 2010 IEEE Second International Conference on Cloud Computing Technology and Science*, CLOUDCOM '10, pages 409–416, Washington, DC, USA, 2010. IEEE Computer Society.

[16] J. Watson. VirtualBox: bits and bytes masquerading as machines. *Linux J.*, Feb. 2008.

[17] A. J. Younge, R. Henschel, J. Brown, G. von Laszewski, J. Qiu, and G. C. Fox. Analysis of Virtualization Technologies for High Performance Computing Environments. In *The 4th International Conference on Cloud Computing (IEEE CLOUD 2011)*, July 2011.