Modern HPC cluster virtualization using KVM and Palacios

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

***Abstract*—In this paper we explore the potential of virtualization being applied to High Performance Computing (HPC). We demonstrate the importance of proper NUMA architecture emulation when running HPC task inside virtual machines on multiple NUMA hosts. We assess KVM/QEMU and Palacios hypervisors and, with proper tuning of hypervisor (including NUMA emulation), we reduce the performance degradation from 10-60% to 1-5% on many tests from HPC Challenge and NAS Parallel Benchmark suites. All tests are performed on modern HPC cluster with high-speed Infiniband interconnect. The cluster nodes are 2-socket 12-core systems, up to 8 nodes were used for computation.**

Comparing KVM/QEMU and Palacios hypervisors, we conclude that in general the results with NUMA emulation enabled 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. We believe that the main advantage of Palacios with respect to performance is the reduced amount of noise generated by the virtualization system. This advantage is getting more important when the scale of the system grows.

I. Introduction

Virtualization technologies are getting more mature and their capabilities have grown rapidly in the last few years. Container-based and 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 [1], [2]. 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 [1]. Nevertheless, serious lack of experimental data still exists. 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.

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

In this work we try to achieve maximum performance of virtualized HPC application compared to the native case. 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) [3] and Palacios hypervisor [4], 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 [5] and NAS Parallel Benchmarks (NPB) [6] as a test suite since these benchmarks are widely used as HPC system performance indicators and HPCC is the basis for building Top 500 Supercomputers list.

Performance tests were made on a modern HPC cluster containing 8 Intel Xeon nodes (96 total processor cores) connected to a 40 Gb/s Infiniband network. Many previous results were gathered on single node systems, which does not fully evaluate virtualization overhead, especially for large-scale parallel applications.

Our main contributions are the following:

- We describe and evaluate some basic but effective ways for tuning KVM/QEMU when running HPC applications. We highlight the importance of NUMA emulation inside VM corresponding to the real NUMA configuration.
- We provide virtualization overhead test results for modern HPC cluster with high-speed Infiniband interconnect. Cluster nodes are 2-socket 12-core NUMA servers, up to 8 nodes are used.
- We compare the performance of Kitten HPC OS + specially crafted HPC hypervisor Palacios 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. In particular, we explore the importance of communication granularity and interrupt rate for overhead.

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

II. 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. [1]. 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 [7], KVM, VirtualBox [8]. Available features, usability and performance are compared. The authors used HPCC and SPEC OpenMP benchmark suites, performance loss for High Performance Linpack (HPL) compared to native run is about 30% on 8-core system. Other test results show overhead varying from a few percent to 50%. As our research shows, this overhead can be mostly eliminated. The authors note that Xen performance is unstable, which greatly decreases scalability of parallel applications. Our preliminary experiments with Xen also showed that it is not well-suited for HPC tasks, at least, without tuning. 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. Scaling of some HPC application may degrade due to the increased noise of virtualized system.

The authors of [1] 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. In general, KVM seems to be more promising, that's why we decided to use it as a basis in our research. On the other side, we cannot claim that Xen has worse performance than KVM. The problem is that every hypervisor together with host OS must be tuned to run HPC application, and we did not perform such tuning for Xen.

Regola and Ducom evaluate the I/O performance of KVM, Xen, OpenVZ [9] and Amazon's EC2 "Cluster Compute Node" [10]. The authors note that CPU virtualization performance is already studied well and that different virtualization approaches provide near-native CPU performance when running CPU-intensive task. On the other hand, 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 gathered on 4-node 32-core system with Xeon E5520 CPUs and Infiniband HCAs passed into VM 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, or it has evolved significantly since 2010, because our results are much 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 tend to 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 [4], the main reason for performance overhead of device passed inside VM is the interrupt overhead. On every interrupt, CPU has to transfer control from VM to the hypervisor, which passes control to the host OS, which determines that this interrupt belongs to VM and passes it back 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 [11], which allows to pass interrupts inside VM without exit to the hypervisor and the host OS. The authors' experimental 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 10 Gbit Ethernet card. With ELI, number of VM exits per second decreases at 100 times on average.

Recent studies show that a virtual machine running over NUMA architecture has significant performance degradation [12] (about 50% on NPB workload B and C class). The NUMA support in current virtualization solutions is incomplete, thus a guest OS can't provide data locality when running on multi-socket hardware. Moreover, NUMA support in hypervisor have a little potential to improve guest data locality. To solve the problem of performance degradation the authors proposed to use paravirtualization. In particular, the authors added new bit transport protocol called "bypass" into OpenMPI implementation. The "bypass" protocol allows communication between VMs using shared memory, visible by participating VMs. When running one VM per one socket, the authors have obtained results which are comparable with UMA systems (5-7% performance degradation). In our work, we performed tests on two-socket 12-core NUMA systems, but our results show that overhead is much less than 50%.

In case of HPC virtualization, hypervisor can emulate NUMA hardware and it will eliminate the NUMA problem since we give all available resources to VM and the VM itself will be responsible for NUMA-aware memory management.

Virtualization of a large scale supercomputer was studied by Lange et al. [4] for the first time using a specially crafted Palacios hypervisor together with Kitten HPC OS [13] used as a host OS. Palacios hypervisor is developed as a part of the V3VEE project [14]. 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. As our research shows, modern multi-socket systems provide new challenges for hypervisors, including correct NUMA support inside the VM. The authors also investigate different paging modes, including shadow paging and hardware-assisted nested paging. It turns out that nested paging behaves better on workloads with extensive page table exchanges, such as Linux. In our work we also prefer to use nested paging since we run Linux as a guest OS.

The authors of Palacios also note the importance of OS noise problem. The impact of OS noise on parallel applications is being widely studied [15], [16]. 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, particularly a large scale one, 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 applications.

III. Virtualization systems under consideration

The most widespread virtualization systems, in general, provide almost the same functionality. Such systems implement most of the hardware-assisted virtualization extensions. 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, since we believe that KVM is more promising and it is much simpler to use. It requires much more time to install Xen and to evaluate it since there are a lot of versions and distributions of Xen. In the future work we hope to find enough time to evaluate Xen too. Also we studied the Palacios hypervisor, which is created specially for computer architecture research and use in high performance computing.

KVM/QEMU

KVM hypervisor is a full virtualization solution for Linux on x86_64 architecture which supports hardware-assisted virtualization extensions (Intel VT-x and AMD-V). KVM consists of several loadable kernel modules and provides special device file for VM management, which can be accessed with `ioctl` system call.

KVM provides only hardware-virtualized CPUs. The rest of the VM is usually implemented by QEMU emulator. QEMU manages resources and implements virtual devices. QEMU can run VMs using hardware-assisted virtualization (provided by KVM) 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 real device address space into guest physical address space using hardware page tables. Without such a device, guest OS should be aware of its placement in real physical memory to issue correct Direct Memory Access requests to assigned devices.

To allow guest system to use high-speed interconnect, we pass real Infiniband HCA into VM using Intel's VT-d. It should be noted that assigned device may perform worse than it performs on the host system. As described earlier in section II, this can be caused by interrupt handling overhead. We elaborate on this problem in section IV. Also we faced the problem associated with virtual BIOS. With Infiniband device passed into VM, at some moment BIOS stops booting the system. To solve this problem, we used the hotplug mechanism and added the device using QEMU CLI after the guest OS had booted.

Use of large pages: The x86_64 virtual memory system has support for different page sizes, including 4KB, 2MB and 1GB pages. 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. Unlike Transparent Hugepages, HugeTLBfs uses reserved huge pages. HugeTLBfs is a special file system. Any file in HugeTLBfs uses huge pages. Custom programs can map files (using `mmap` system call) from HugeTLBfs and these mappings will be backed by huge pages.

The version of QEMU which we use (1.0) allocates memory for VM via the call to `posix_memalign` with 2MB alignment, thus Transparent Hugepages mechanism should work 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 already has support for NUMA system emulation, but emulated structure does not correspond to the real hardware topology. To solve this problem we patched QEMU. Firstly, we pin virtual CPUs to distinct physical cores to disallow QEMU's threads migration between cores. Secondly, we used `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. HugeTLBfs has some problems with security and scalability [17], but it is not significant in our case.

Palacios

The Palacios hypervisor [4] was developed with the goal to effectively virtualize HPC applications. The 1.0 release became available at 2008, and the latest 1.3 release appeared at November, 2011. The distinguishing feature of Palacios is its embeddable nature achieved using a set of unified host OS interfaces. These interfaces could be implemented in the host OS, and together with Palacios binary they provide hypervisor functionality to the host OS user. Initially Palacios was embedded into Kitten HPC OS, but in the latest 1.3 release Linux support is implemented too. Another important feature of Palacios is the possibility to configure it in a lot of ways at compile time, to include or exclude different modules in the resulting binary. This feature allows for simple creation of a set of hypervisors suitable for different needs, including experimental use.

We decided to experiment with 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 [4].

Device assignment technique: The authors of Palacios note that the real high-speed communication device should be passed into VM to achieve performance which is comparable to the native run. To make device assignment possible, special paravirtual interface is used (the whole interaction between guest system and hypervisor is called *Symbiotic Virtualization*). 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 with simple offset addition. In fact, changes required in guest OS to implement this interface are relatively simple. We created our own patch initially for 2.6.18 Linux kernel and then ported it to the newer kernel versions.

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 a lot of difficulties caused by its immaturity. They include incomplete device assignment support (PCI Capabilities were not passed into VM, causing MSI, MSI-X interrupts not to work), incomplete Intel VT-x support, maximum $\sim$ 3.5GB of virtual RAM, no NUMA support and a lot of small bugs. 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.

IV. Performance evaluation and discussion

We tried to cover a wide range of HPC applications using two popular benchmark suites — HPCC and NPB — 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 used an HP cluster as a testbed. Cluster consists of 8 HP ProLiant BL2x220c G7 blades, each blade combines two server nodes. For tests we used up to 8 nodes. Each node contains 2 Intel Xeon X5670 2.93GHz CPUs (6 cores per CPU) and 24GB of RAM. Hyperthreading was disabled on all nodes. Nodes communicate via 1Gbit/s. service Ethernet network, and 40Gbit/s. Infiniband network used for computing.

The system used for native tests and as a guest system is Linux CentOS 6.0 with kernel `2.6.32-71.29.1.el6.x86_64`. The kernel is modified to support Palacios device assignment. MPI library is OpenMPI 1.4.3, GNU Compiler Collection version used to build test suites is 4.4.4. Infiniband stack is provided by Mellanox OpenFabrics Enterprise Distribution for Linux, version `1.5.3-1.0.0-rhel6-x86_64`.

The system used as a KVM/QEMU host is Linux CentOS 6.2 with kernel `2.6.32-220.2.1.el6.x86_64`. QEMU version is `qemu-kvm-1.0` with our modifications which were described earlier. Palacios version is based on our own local branch, our changes are described in section III. 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. Infiniband HCA is passed inside VM as mentioned earlier. For Palacios, we also pass Ethernet card and SATA controller.

[1]We contributed our code to the V3VEE project, some of our patches could be found at the V3VEE project Web site [14] and some are already integrated into the development branch.

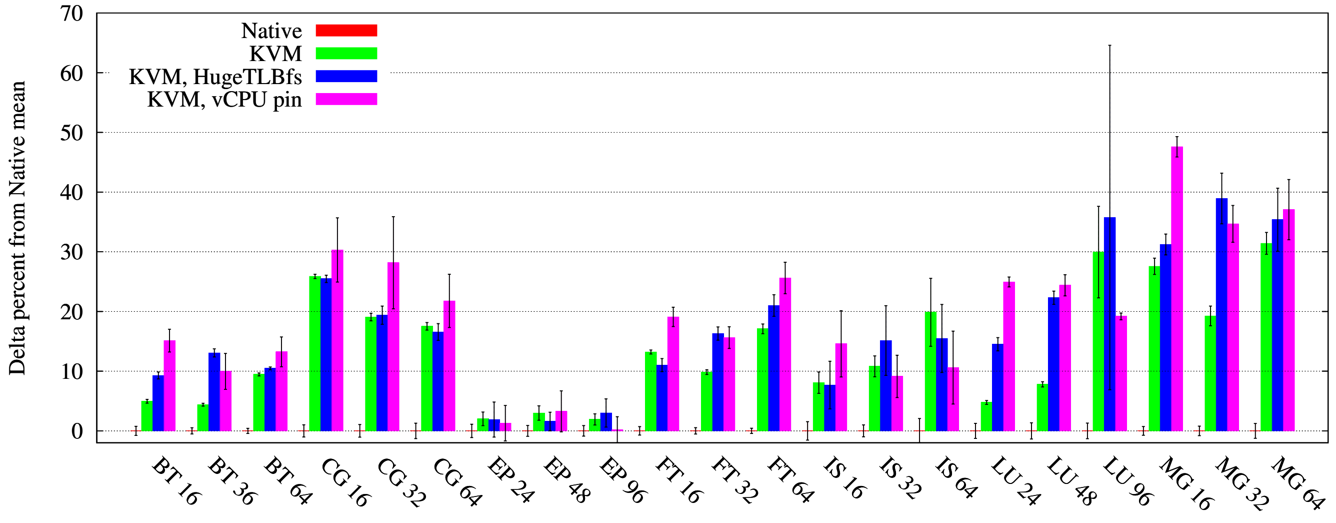

Fig. 1. Initial NPB results for KVM/QEMU.

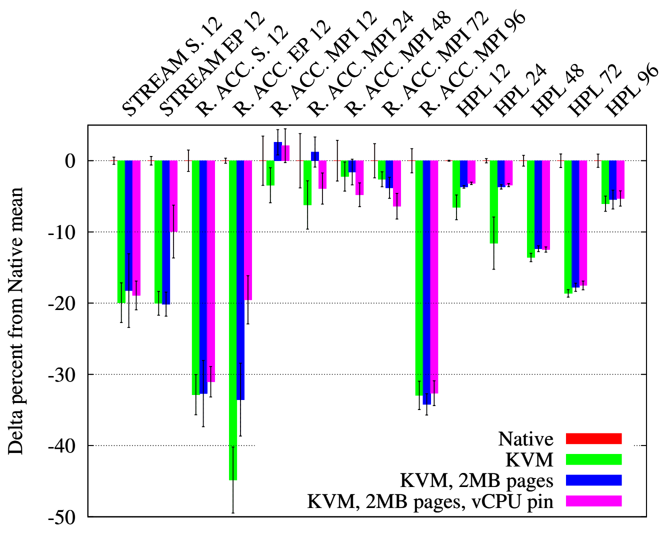

Fig. 2. Initial HPCC results for KVM/QEMU.

Benchmarks: To measure the performance we use HPC Challenge and NAS Parallel Benchmark suites. We run each benchmark on 2, 4 and 8 nodes. NPB version is 3.3.1 for MPI (NPB3.3-MPI). From NPB suite we employ IS, EP, CG, MG, FT, BT and LU tests with the problem class C.

IS stands for Integer Sorting, and performs a lot of communication and random memory accesses. EP is Embarrassingly Parallel Gaussian random variates generator. CG is a program using Conjugate Gradient method, with irregular memory accesses and communication. MG (MultiGrid) approximates the solution to discrete Poisson equation, provides long and short-distance communication, memory intensive. FT is discrete 3D fast Fourier Transform with all-to-all communication. The BT and LU are solvers for systems of partial differential equations using Block Tri-diagonal and Lower-Upper Gauss-Seidel solvers.

For IS, CG, FT and MG tests we run 8 processes per node (number of processes must be a power of two), for EP and LU we run 12 processes per node (no restrictions), for BT run use 8, 9 and 8 processes per node (number of processes must be square). NPB benchmark is also executed on 1 node with 9 processes for BT, 12 for EP and LU and 8 for other tests. The results of NPB suite tests 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. The PTRANS test uses too small amount of time (0.1-0.5 sec. on average) and it belongs to the same class of problems as the STREAM benchmark. We also exclude DGEMM, Bandwidth/Latency, FFT benchmarks' results due to the space limitations. Please email us to get the full test results. Selected tests from HPCC suite have the following characteristics:

- **STREAM.** Measures sustainable memory bandwidth for four vector kernels (Copy, Scale, Add, Triad), we assess Triad results (in Gbytes/sec.). 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. We use MPI, Single and EP versions.
- **HPL** (High Performance Linpack). 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.

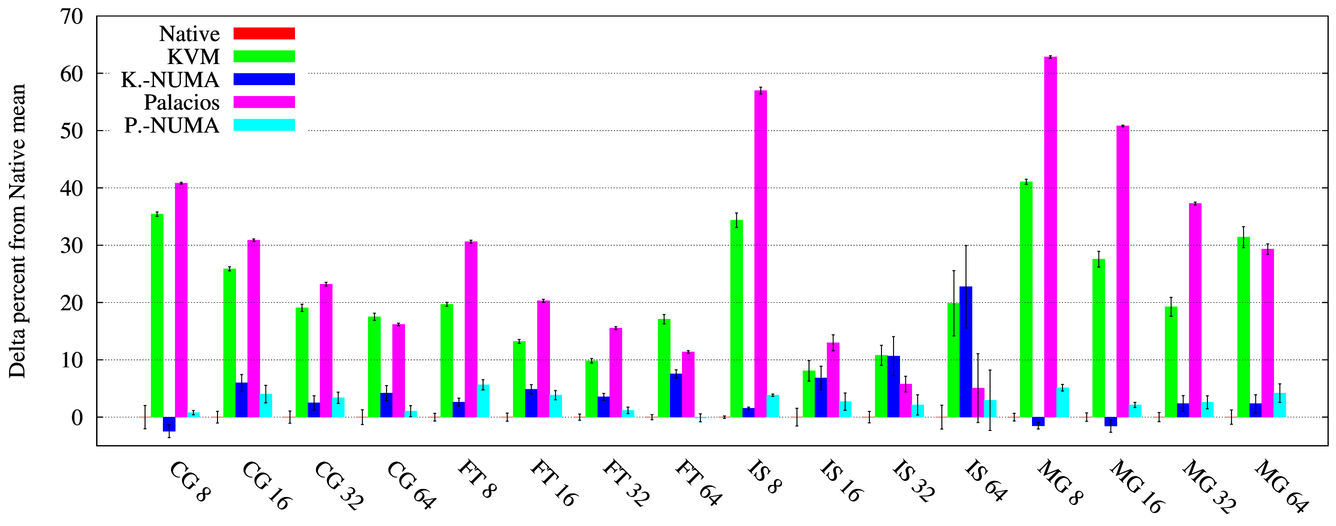

Fig. 3. KVM/QEMU and Palacios results for NPB CG, FT, MG, IS tests.

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 and HPCC runs 20 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 12 virtual CPUs per VM and default memory allocation mechanism. Initial results for NAS Parallel Benchmarks are presented in Figure 1, with "KVM" meaning the default configuration and 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 corresponding test name and process count used for computation. As the Figure shows, in the "KVM" configuration performance overhead ranges from 2-4% on EP test to 32% on MG test for 64 processes.

Next we assumed that QEMU virtual CPU threads migration between physical CPU cores is crucial for virtualization overhead. To disallow threads migration, we set the affinity for each of QEMU's CPU threads to the corresponding physical core, one virtual CPU per physical core. Also we checked that HugeTLBfs mechanism provides the same overhead as Transparent Hugepages. The NPB results for KVM with these changes are provided in Figure 1, bars named "KVM, vCPU pin" and "KVM, HugeTLBfs". Note that these two groups of tests were performed only 10 times due to the time constraints. As the Figure shows, we got the same or even much greater performance overhead after these changes, with vCPU pin case having more unstable results on BT, CG and FT tests. It could be caused by non-local memory access overheads, when CPU from one node makes accesses to memory from another node. Also note the huge unstability of HugeTLBfs case on the LU test with 96 processes. It seems that some unaccounted factor influenced this case a lot.

Initial KVM/QEMU results for HPCC benchmark are provided in Figure 2. In this case, we assess configurations with HugeTLBfs and virtual CPU pinning together with HugeTLBfs. "R. ACC." stands for Random Access, "S." is a Single version of the test. Note that the delta percent from Native mean is negative since in this case the results are in Gbytes/sec. (STREAM), GUP/sec. (Random Access) and Gflops (HPL), so the overhead causes these values to decrease. Stream Single and EP overhead is around 20% except vCPU pin case, where we get only 10% on EP. Random Access shows noticeable overhead for Single and EP versions — from 20 to 45 %, and again, vCPU pin case is the best on EP with 20% overhead. Random Access MPI version results are unstable and overhead is less than 7% when running on up to 72 processes, but for 96 processes the picture drastically changes — we see persistent overhead of about 34%. It is possible that communication becomes the bottleneck on 96 processes and as a result overhead grows, but we did not explore this issue in details.

Finally 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.

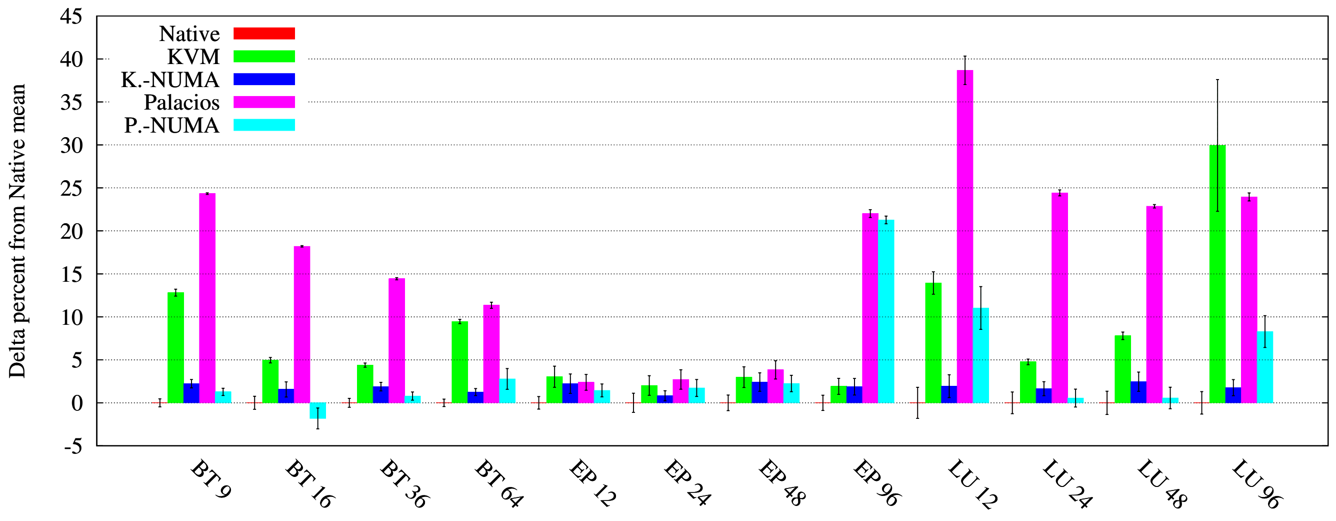

Fig. 4. KVM/QEMU and Palacios results for NPB BT, EP, LU tests.

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 two configurations (default and NUMA) of KVM/QEMU and Palacios are provided in figures 3 and 4. Results for HPCC suite are in Figure 5.

The first obvious 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 huge 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 emulation 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. KVM behaves better when running test on 24, 48, 72 processes, while Palacios is better on 96 processes.

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. The case with this test is really unexpected since it only depends on CPU performance. 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. This hypothesis is partially confirmed by the fact that both vCPU pinning and use of HugeTLBfs decrease the performance breaking the default Linux process and memory management mechanisms as Figure 1 shows.

We can also see that sometimes the results inside VM with NUMA support are better than native. For example, look at tests CG 8, MG 8, MG 16, where KVM with NUMA behaves better than native, and BT 16, where Palacios with NUMA is better than native. Probably the reason for this behavior is that some unaccounted and persistent factors were present during these runs, but more thorough investigation is required.

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% (IS 64 case). Probably this advantage is associated with the decreased noise of Kitten OS when compared to Linux, especially for fine-grained IS test. We also investigate the granularity of communications in NPB tests by evaluating the interrupt rate, see the next subsection.

If we compare NUMA results of KVM/QEMU with Palacios and use 3% difference as a threshold, we find that KVM is better in 11 cases, Palacios is better in 9 cases, and results are almost the same in 22 remaining cases. With 5% difference used as a threshold, KVM is better in 8 cases while Palacios only in 4 cases. In general, KVM provides more stable and predictable results, while Palacios is much better on fine-grained tests like IS (especially with large process count). At the same time, Palacios shows abnormal performance degradation on some tests.

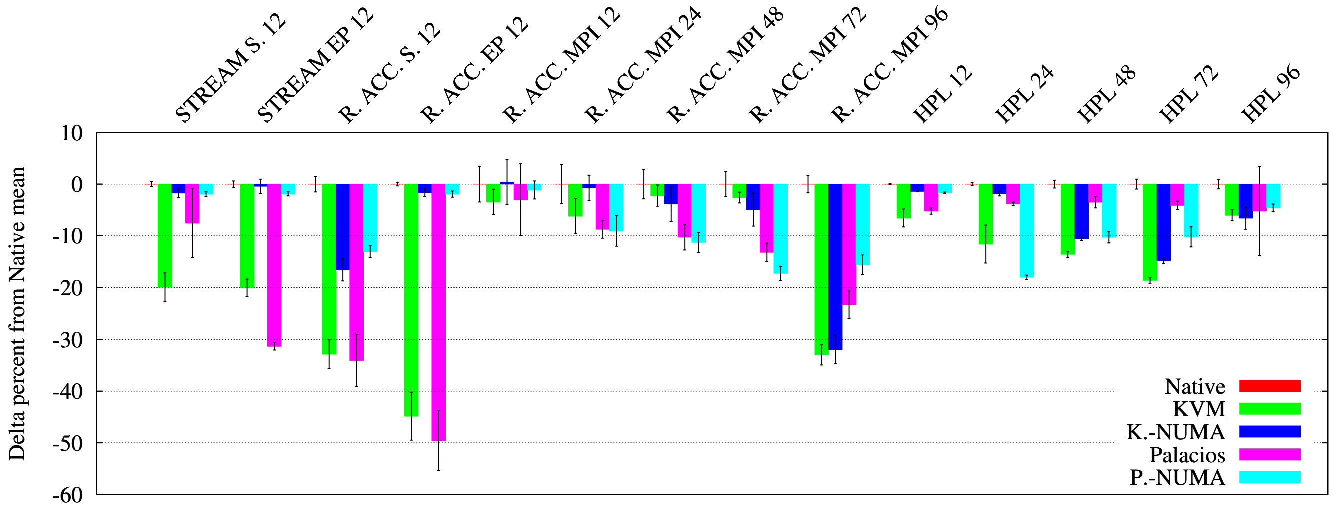

Fig. 5. KVM/QEMU and Palacios results for HPCC suite.

Interrupt rate influence: Interrupt virtualization could cause huge performance overhead when using assigned devices [11]. To check this theory in our case, we measure the Infiniband interrupt rate during the execution of NPB tests. The results are presented in Figure 6. There are two data series combined: overhead percent relative to the native case is in red and average interrupts per second (IPS) count for Infiniband device is in green.

Maximum interrupt rate is almost 11000 IPS, meaning at least 11000 VM exits per second which should cause noticeable performance overhead. Though in the paper [11] interrupt rate was higher (40000-60000 IPS in baseline case), we believe that in our case at least some part of present 20% overhead on IS test is due to the high interrupt rate. To prove this assumption, we should run tests with some analog of ExitLess Interrupt system enabled.

High interrupt rate for IS test with 32 and 64 processes used also reflects the granularity of communications, in this case the communication seems to be fine-grained. Figure 3 shows that Palacios' overhead is 20% smaller than KVM's. As it was noted before, this difference may be associated with the decreased noise of the Kitten OS.

V. Conclusion

Our primary contribution has been to demonstrate the importance of NUMA architecture emulation according to the real configuration when running HPC task inside virtual machines on multiple NUMA hosts. In particular, we explored KVM/QEMU and Palacios hypervisors. KVM is widely used in industry while Palacios is 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. The only tests which caused problems are Random

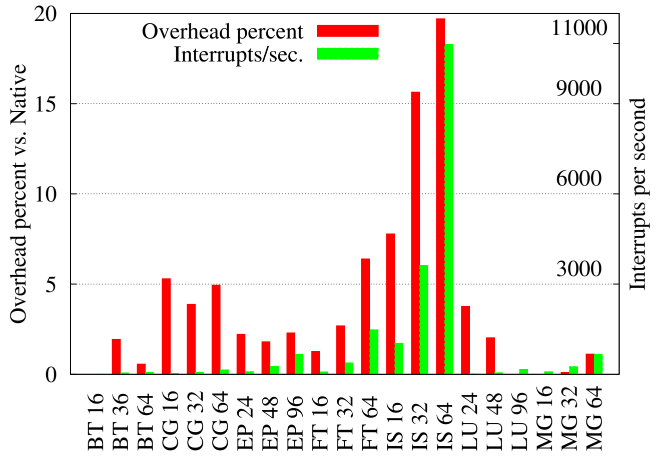

Fig. 6. NPB Overhead vs. Interrupts per sec. for KVM-NUMA case.

Access and HPL tests from HPCC suite, with performance overhead up to 30% for the former and 15% for the latter.

All tests were performed on modern HPC cluster with high-speed Infiniband interconnect. Up to 96 processor cores were used for computation. Gathered test results allowed to evaluate both CPU and communication overheads caused by virtualization. We also explored the correlation of communication device interrupt rate and virtualization overhead; for our Infiniband device, maximum interrupt rate was almost 11000 interrupts per second on IS test from NPB suite running on 64 processes and corresponding overhead for KVM/QEMU was around 20%. Though we cannot claim that the overhead is caused only by this interrupt rate, it can be considered as an evidence of IS test being fine-grained.

Concerning KVM/QEMU and Palacios comparison, we 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 at large scale, 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 multi-core, multi-socket HPC systems.

REFERENCES

[1] 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.

[2] 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.

[3] 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*, Jul. 2007, pp. 225–230.

[4] 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*, ser. VEE '11. New York, NY, USA: ACM, 2011, pp. 169–180. [Online]. Available: http://doi.acm.org/10.1145/1952682.1952705

[5] 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*, ser. SC '06. New York, NY, USA: ACM, 2006. [Online]. Available: http://doi.acm.org/10.1145/1188455.1188677

[6] 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.

[7] 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.*, vol. 37, pp. 164–177, Oct. 2003. [Online]. Available: http://doi.acm.org/10.1145/1165389.945462

[8] J. Watson, "VirtualBox: bits and bytes masquerading as machines," *Linux J.*, Feb. 2008. [Online]. Available: http://dl.acm.org/citation.cfm?id=1344209.1344210

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

[10] 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*, ser. CLOUDCOM '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 409–416. [Online]. Available: http://dx.doi.org/10.1109/CloudCom.2010.71

[11] 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).

[12] K. Z. Ibrahim, S. Hofmeyr, and C. Iancu, "Characterizing the performance of parallel applications on multi-socket virtual machines," in *Proceedings of the 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, ser. CCGRID '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 1–12. [Online]. Available: http://dx.doi.org/10.1109/CCGrid.2011.50

[13] 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)*, April 2010, pp. 1 –12.

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

[15] 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.*, November 2008, pp. 1 –12.

[16] 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*, November 2003, p. 55.

[17] A. Arcangeli, "Transparent Hugepage Support, http://www.linux-kvm.org/wiki/images/9/9e/2010-forum-thp.pdf," *KVM Forum 2010*, 2010.