Journal of Physics: Conference Series **219** (2010) 042005 doi:10.1088/1742-6596/219/4/042005

A quantitative comparison between xen and kvm

Andrea Chierici

Riccardo Veraldi

INFN-CNAF

{chierici,veraldi}@cnaf.infn.it

Abstract. Virtualization is a proven software technology that is rapidly transforming the IT landscape and fundamentally changing the way that people compute. Recently all major software producers (e.g. Microsoft and RedHat) developed or acquired virtualization technologies. Our institute is a Tier1 for LHC experiments and is experiencing lots of benefits from virtualization technologies, like improving fault tolerance, providing efficient hardware resource usage and increasing security. Currently the virtualization solution adopted is xen, which is well supported by the Scientific Linux distribution, widely adopted by the HEP community. Since the HEP linux distribution is based on RedHat ES, we feel the need to investigate performance and usability differences with the new kvm technology recently acquired by RedHat. The case study of this work will be the LHCb experiment Tier2 site hosted at our institute, where all major grid elements run on xen virtual machines smoothly. We will investigate the impact on performance and stability that a migration to kvm would entail on the Tier2 site, as well as the effort required by a system administrator to deploy the migration. Several quantitative test results will be shown and explained in detail.

1. Introduction

Infrastructure virtualization, and in particular server virtualization has become very important nowadays because of the tremendous benefits achieved by using it. Virtualization is a technology that allows running a certain number of different and concurrent operating system instances inside a single physical machine. This physical server is divided into multiple isolated virtual environments called guests.

2. Virtualization approaches

There are three popular approaches to server virtualization: full virtualization, para-virtualization and virtualization with hardware support (Hardware Virtual Machine, or HVM).

2.1. Full virtualization

Virtual machines are based on the host/guest paradigm where each guest runs on a virtual imitation of the hardware layer. This approach allows the guest operating system to run without modifications. It also allows the administrator to create guests that use different operating systems. The guest has no knowledge about the host operating system because it is not aware that it's not running on real hardware. It does, however, require real computing resources from the host, so it uses a *hypervisor* to coordinate instructions to the CPU. The hypervisor is called a virtual machine monitor (VMM). It validates all the guest-issued CPU instructions and manages any executed code that requires additional privileges. VMware and Microsoft Virtual Server both use the full virtualization approach.

2.2. Para-virtualization

The para-virtualized machine approach (PVM) is based on the host/guest paradigm and uses a virtual machine monitor. In this model, however, The VMM actually modifies the guest operating system's code; this modification is called porting. Porting supports the VMM so it can access privileged systems calls sparingly. Like virtual machines, para-virtual machines are capable of running multiple operating systems. Xen and UML both use the para-virtual machine model. The main advantage of this approach is the speed that can be achieved, always faster than HVM approach.

2.3. Hardware virtual machine (HVM)

Recent innovations in hardware, particularly in CPU, MMU and memory components (notably the Intel VT-x and AMD-V architectures), provide some direct platform-level architectural support for OS virtualization.

HVM offers one key features: it avoids the need to trap and emulate privileged instructions by enabling guests to run at their native privilege levels. For some hypervisors (like xen and kvm) it is possible to recompile para-virtualized drivers inside the guest machine running in HVM environment and load those drivers into the running kernel to achieve para-virtualized I/O performance within an HVM guest.

3. Xen based virtualization

Xen [1] is a virtual machine monitor that allows several guest operating systems to be executed on the same computer hardware concurrently. A Xen system is structured with the Xen hypervisor as the lowest and most privileged layer. Above this layer are located one or more guest operating systems, which the hypervisor schedules across the physical CPUs. Xen can work both in para-virtualized or HVM mode; in the first the guest operating system must be modified to be executed. Through para-virtualization, Xen can achieve very high performance. The HVM mode offers new instructions to support direct calls by a para-virtualized guest/driver into the hypervisor, typically used for I/O or other so-called hypercalls.

4. KVM base virtualization

KVM [3] is a full virtualization solution for Linux on x86 hardware containing virtualization extensions (Intel VT or AMD-V). It consists of a loadable kernel module, kvm.ko, which provides the core virtualization infrastructure, and a processor specific module, kvm-intel.ko or kvm-amd.ko. KVM requires a modified version of qemu, a well known virtualization software. The kernel component of KVM is included in mainline Linux, as of 2.6.20, while for xen only external support is available. KVM supports I/O para-virtualization using the so called VIRTIO subsystem consisting of 5 kernel modules**.**

4.1. KVM in our centre

KVM can easily be installed using a *yum* repository [4]; once installed the first thing to do is to set up networking. For our virtual machines we currently use public IP addressing configured via a bridged network, so we modified the default kvm start-up script to create a virtual bridge for each network interface; in this way we can choose at boot time which virtual machine can be hooked to which network interface. After this, we need to add a “tap” interface and assign it to the proper bridge: even in this case we created a special script “/etc/qemu-ifup” that is executed automatically upon guest creation and destruction so that tap interfaces are managed transparently. Creating a virtual bridge is the standard method to allow different virtual interfaces belonging to different virtual machines to be able to access the physical network interface managed by the hypervisor. Sometimes this solution is also called “virtual switch”.

5. KVM: Qualitative test

We introduced kvm into our environment seamlessly. We are using Quattor [5] as the main installation tool, which is based on machine profiles that describe the configuration expected for every machine by system administrators. Introducing kvm didn't require any additional effort or modification to this infrastructure, getting every VM to be installed via network boot exactly like xen VMs. We stepped into one little problem involving the *"-boot"* option, which requires only one parameter, being it network, hard disk or cdrom. After the machine has installed via network boot, we have to stop and restart it with the option to boot from hard disk, while with xen this is not needed, being hard disk the second default boot device.

The qualitative test consisted in installing on a kvm VM an "EGEE Computing Element" used in production in the LHCb tier2 we are hosting at CNAF. Before the test, the lhcb tier2 site was running totally on xen VMs: it consisted of 2 Computing Elements installed on 2 different hosts (to balance performance and assure site reliability), a Storage Element and a Monitoring Box; by reinstalling one CE on a kvm VM we had the opportunity to make a direct comparison between the two virtualization technologies. The VM worked flawlessly for more than 3 weeks not giving users and system administrators any idea of the change that was made: we considered the test fully passed.

While we were doing these tests we decided to install also a CMS secondary squid server and we had the same feedback from the users: no differences in performance were noticed. As a reference, we used this hardware to host the two kvm VMs: 1 dual Intel E5420, 16GB ram, sata disks on Areca controller.

6. Quantitative test

Even if kvm passed brilliantly the qualitative test, we need some quantitative measures to confirm the positive impression obtained. For this reason we performed a set of tests targeted to measure the classic parameters of a machine (CPU, network and disk access), with tools well known to the HEP community. Here is a list of the tools used:

- CPU: hep-spec06 (v1.1)
- Network: iperf (v2.0.4)
- Disk access: bonnie++(v1.94)

To better understand the performance of a kvm virtual machine, we tested the same parameters also for a xen machine, both with para and hvm virtualization method and compared them with a non virtualized machine (the baseline).

6.1. Test specifications.

The hardware used for the quantitative test is composed of 1 blade server with a dual intel E5420, 16GB ram and two 10k sas disks connected to a LSI Logic raid controlled (disks configured with raid0 option).

The VMs specs were:

- Xen-para VM: 1 vcpu, 2 GB ram, disk on a file
- Xen-hvm VM: 1 vcpu, 2GB ram, disk on a file, "netfront" network driver
- KVM VM: 1 vcpu, 2GB ram, disk on a file, e1000 network driver emulation

As for the OS installed:

- Host OS: SL 5.2 x86_64, kernel 2.6.18-92.1.22.el5
- VM OS: SLC 4.5 i386, kernel 2.6.9-67.0.15.EL.cern

Version of the hypervisors used:

- KVM: 83
- Xen: 3.2.1

6.2. Benchmarks: HEP-Spec06

HEP-Spec06 [6] is the new standard in HEP community for CPU performance benchmarking and it is based on a subset of the Spec benchmark. We performed a wide number of tests in order to quantify

the differences in performance for the various virtualization solutions, compared to a non virtualized CPU.

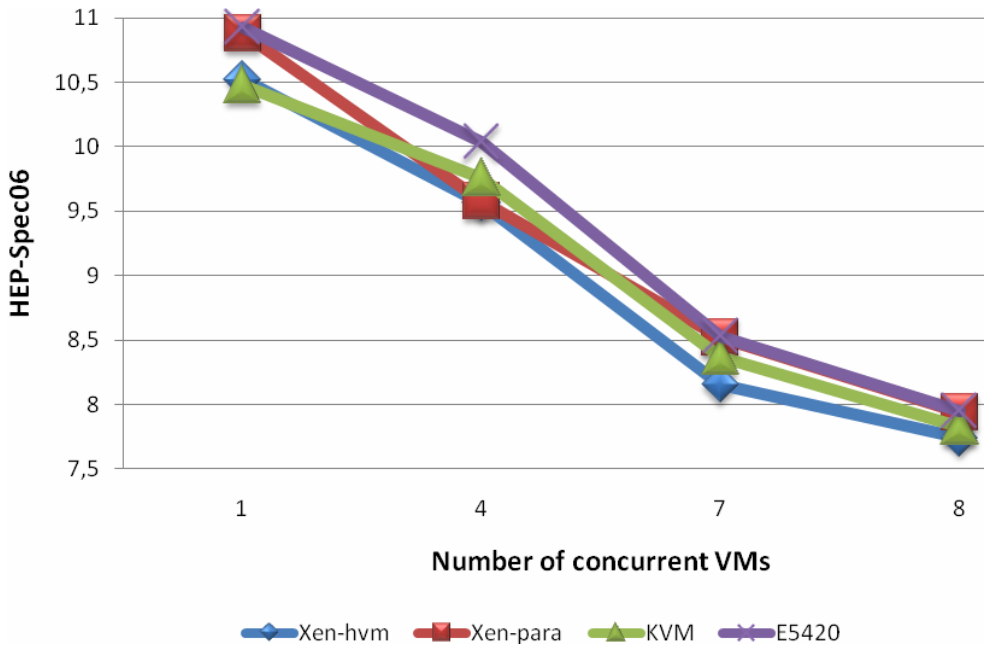

Figure 1: XEN vs. KVM on dual Intel E5420, single performance measure

In figure 1 we show the performance comparison of the various hypervisors while an increasing number of virtual machine runs concurrently on the host. We tested the performance with 7 and 8 concurrent runs in order to understand if the hypervisor requires a dedicated CPU (1 hypervisor + 7 VMs is the exact number of cores on our box).

The results of the test shown in figure 2 confirm that we can “overload” the cores of our machine and run 8 VMs concurrently, without any significant performance loss.

Figure 3 shows a comparison between the HEP-Spec06 benchmark on 8VMs and the same benchmark run on physical CPUs, as calculated by the HEPiX community [7]. Again, as we can see, the performance loss is so little that we can consider it almost zero. The benefits obtained by virtualization, overcome the minimal performance loss. Table 1 shows the exact percentage loss compared to physical CPUs. With virtualization technologies, either xen or kvm, the loss is comparable to a CPU downgrade (to the preceding model).

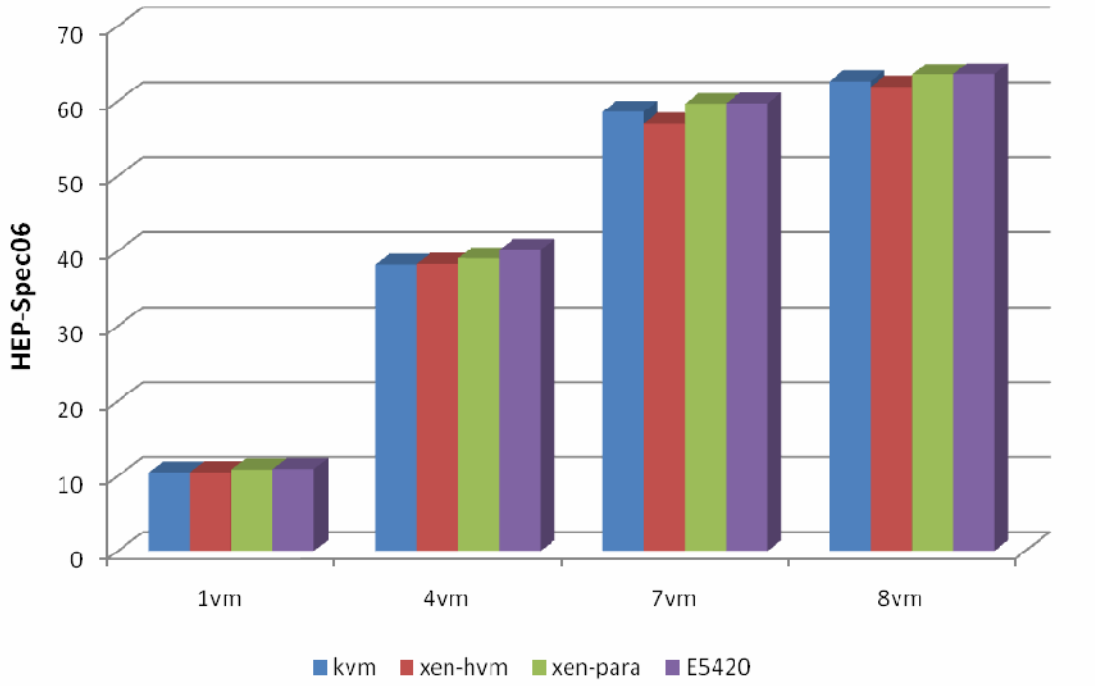

Figure 2: VMs vs. CPU

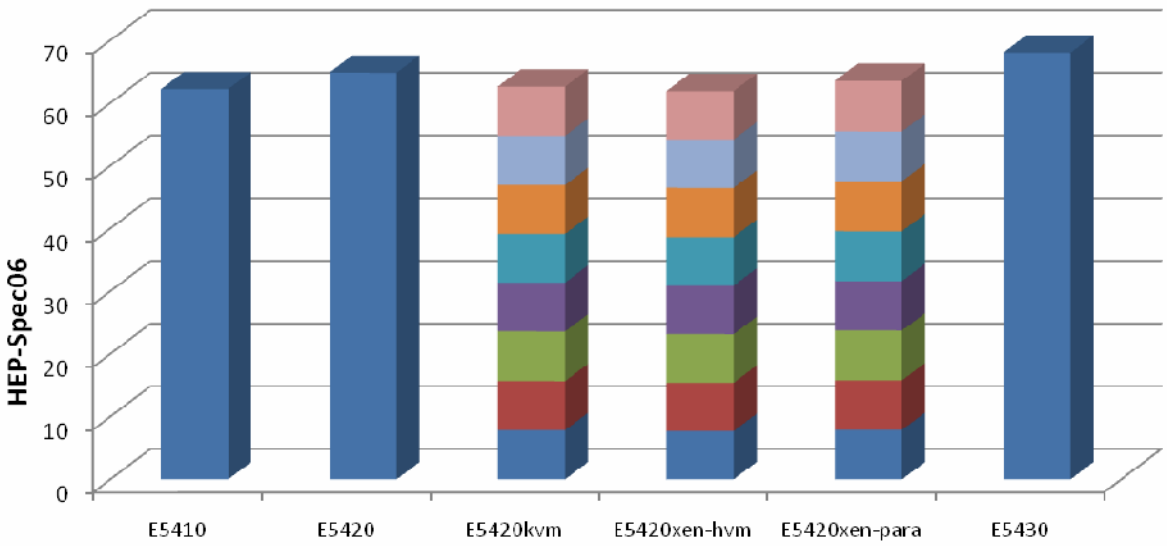

Figure 3: 8VMs aggregate vs. CPUs

Virtualization Technology	**% loss from non Emulated CPU (E5420, 8vm)**
E5420kvm	3,42
E5420xen-hvm	4,55
E5420xen-para	2,02
E5410 vs. E5420	4,07

Table 1: HEP-Spec06 % loss

6.3. Benchmarks: Iperf

Network performance is an essential parameter to measure: with iperf we tested the throughput performance both in inbound and outbound directions. The options we used to get the results are: "-w256k –P 5 –t 900" that means a TCP window size of 256k, 5 parallel connections and the duration of 900 seconds (15 minutes).

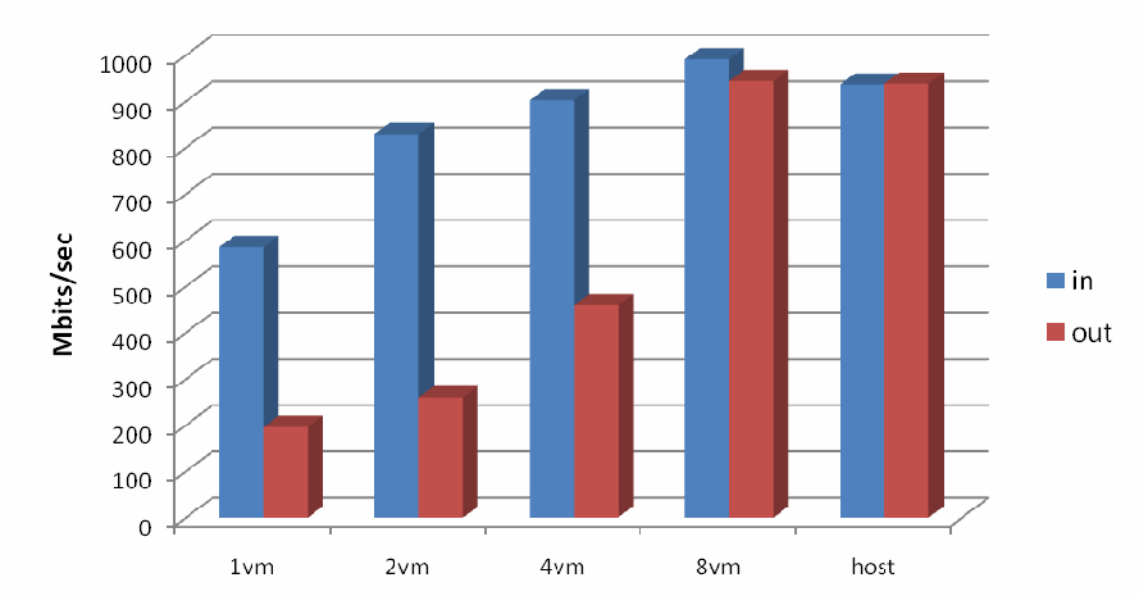

Figure 4: KVM Network Performance

As we can see from figure 4, the behavior of the network is asymmetric with inbound performing better than outbound connection. The situation improves with the increasing number of the nodes, giving us the idea of some sort of limitations hardcoded somehow inside the driver. We are going to better investigate this aspect in the near future.

In figure 5 we can see a global comparison of the network performance: xen is symmetric in every circumstance and so proves to be a better choice in situations where network outbound speed is critical.

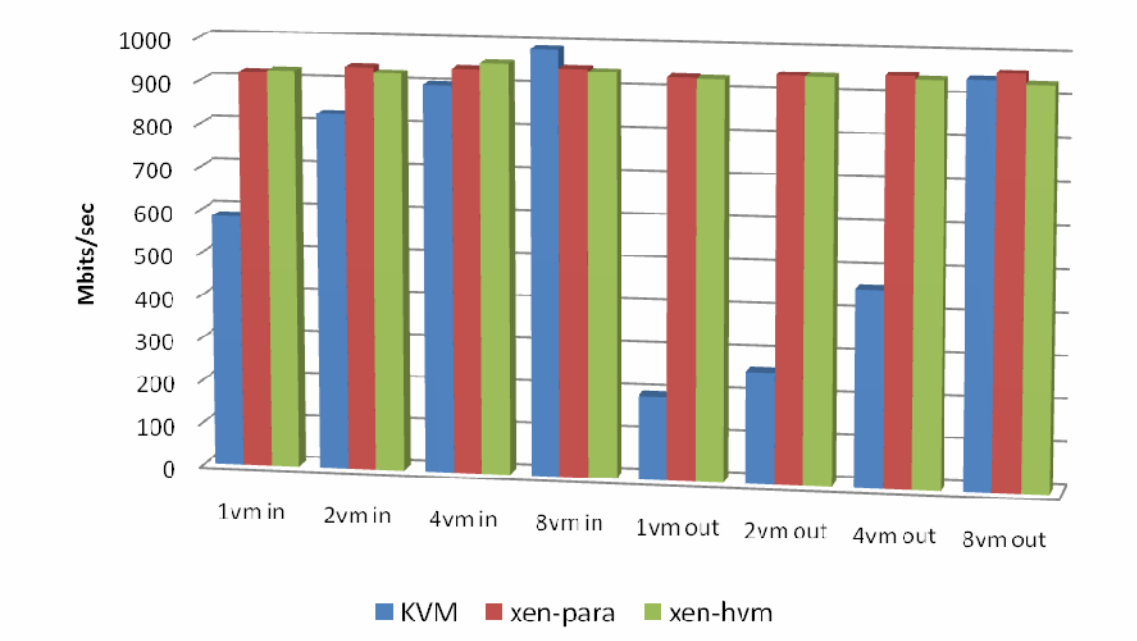

Figure 5: Network performance comparison

6.4. Benchmarks: bonnie++

Disk access speed is critical for every virtualization technology and our tests showed that a lot is still to be improved. In our centre we decided to adopt the disk-on-a-file approach instead of the disk-on-a-partition. This solution, although may limit the I/O performance a little, is more convenient when having to deal with backup and migration of disk images to other hosts. In figure 6 indeed we can see the read/write speed performance for a real machine compared to a virtual one (either xen or kvm). Xen with para-virtualized access is by far the best solution, particularly reading from disk. What really disappointed us are the tests with concurrent access of VMs to the disk. In these circumstances (figure 7 and 8) all the solutions perform very badly compared to the real machine, with a significant advantage to xen-para, even if only for reading performance. All the solutions perform very badly in disk writing, leaving a lot of room for improvement in future releases of the hypervisors.

The next step in hardware virtualization is to implement special instruction in the hardware in a way similar to what has been done for the CPU, that as we have proved, are already performing brilliantly.

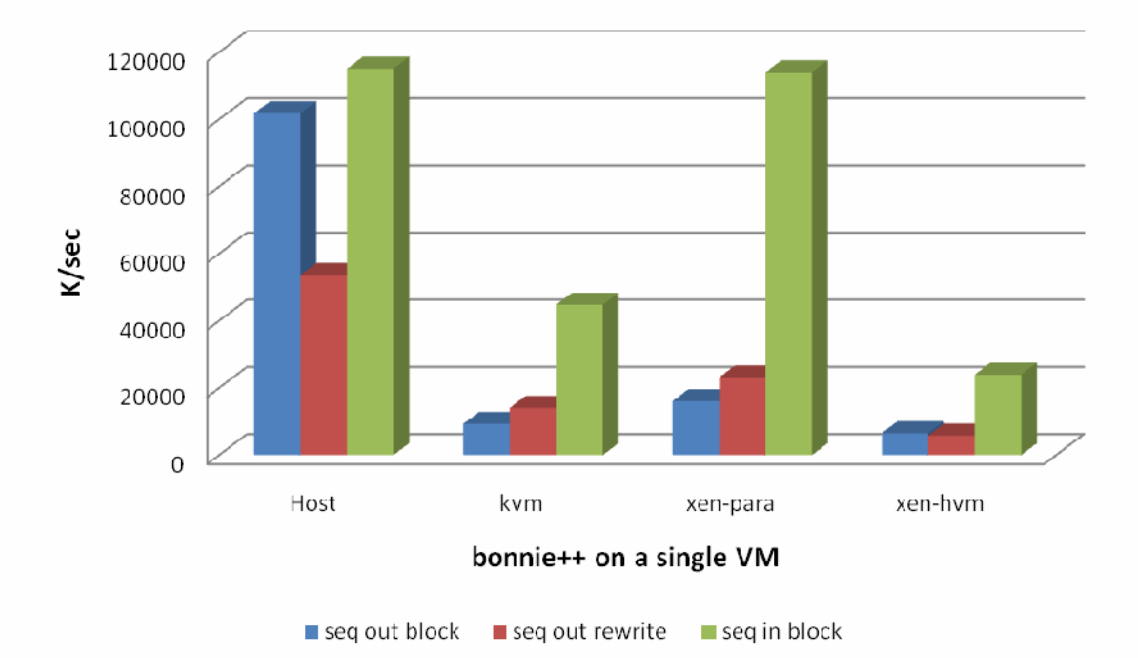

Figure 5: 2GB Ram, 4GB data set, 1vm comparison

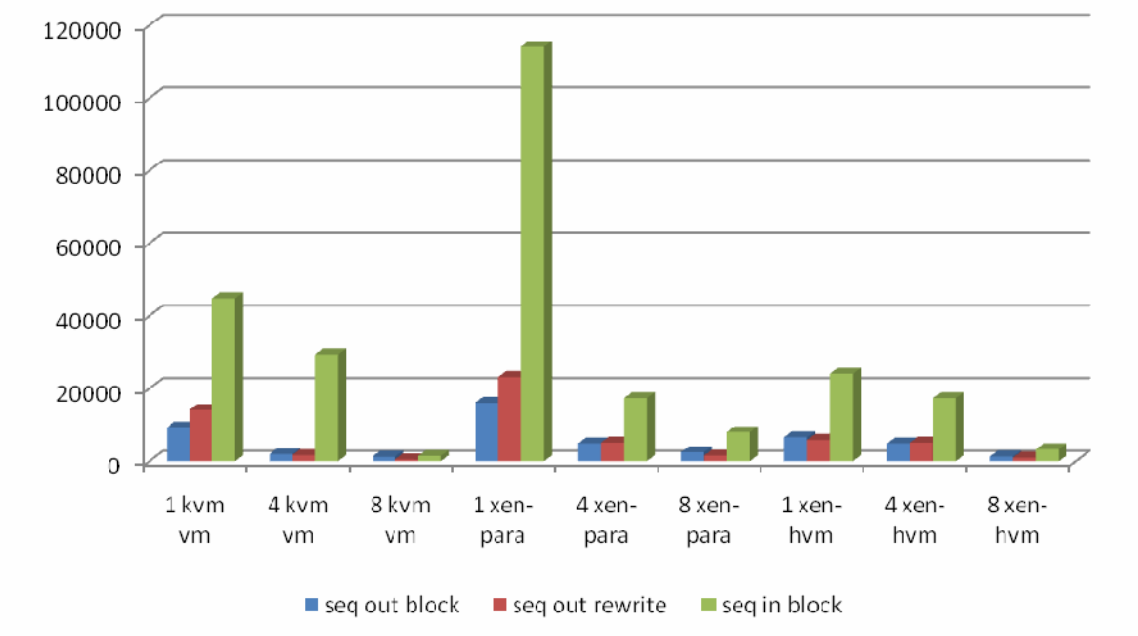

Figure 6: 2GB ram, 4GB data set, 8vm, single

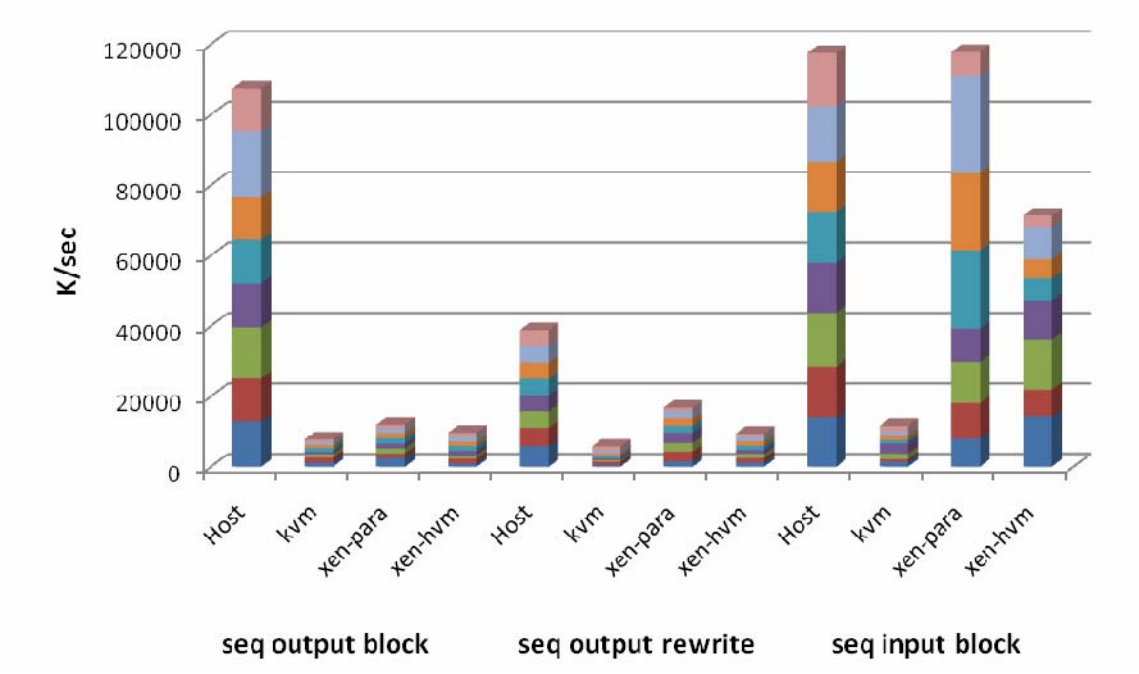

Figure 7: 2GB ram, 4GB data set, 8vm, aggregate

7. Conclusions and future work

During our tests kvm proved great stability and reliability: it never crashed and integrated seamlessly into our computing farm, without requiring any additional effort to the system administrators.

Our benchmarks showed that the CPU performance provided by the virtualization layer is comparable to the one provided by xen and in some cases it's even better.

Network performance is fair, showing some strange asymmetric behavior, but anyway we consider them acceptable.

Disk I/O seems to be the most problematic aspect, providing the VM poor performance, particularly when multiple machines concurrently access the disk. Anyway even xen based VM showed poor performance with this parameter, maybe caused by the solution adopted in our center, that is to provide virtual disks on a file.

To summarize, we can say that even if looking very promising, right now, xen hypervisor seems to be the best solution, particularly when using the para-virtualized approach.

These tests have been very useful for us and gave some input on future investigations and tests to perform: we particularly want to test kvm "virtio" drivers (only available in latest linux kernels, not suitable for EGEE middleware), and to test the qemu snapshot features for creating backups. I/O performance is another critical aspect that deserves more attention: we want to test performance using disk partitions instead of disk-on-a-file approach and to investigate the impact on performance by storing the virtual disks on different physical media.

References

[1] Xen: http://www.xen.org/
[2] Xen unofficial repository: http://www.gitco.de/repo
[3] Kvm: http://www.linux-kvm.org
[4] Kvm unofficial repository: http://www.lfarkas.org/linux/packages/centos
[5] Quattor: http://www.quattor.org
[6] HEP-SPEC06: https://twiki.cern.ch/twiki/bin/view/FIOgroup/TsiBenchHEPSPEC
[7] HEPiX: http://www.hepix.org