Performance improvements in a large scale virtualization system

Davide Salomoni[*] **INFN-CNAF**
E-mail: davide.salomoni@cnaf.infn.it

Anna Karen Calbrese Melcarne INFN-CNAF
E-mail: anna.karen.melcarne@cnaf.infn.it

Andrea Chierici INFN-CNAF
E-mail: andrea.chierici@cnaf.infn.it

Gianni Dalla Torre INFN-CNAF
E-mail: gianni.dallatorre@cnaf.infn.it

Alessandro Italiano INFN-CNAF
E-mail: alessandro.italiano@cnaf.infn.it

This work shows the optimizations we have been investigating and implementing at the KVM virtualization layer in our research institute based on more than a year of experience in running thousands of VMs in a production environment used by several international collaborations. These optimizations increase the adaptability of virtualization solutions to demanding applications like those run in our institute (High-Energy Physics).
In particular we will show what is the best solution to share storage space from an hypervisor to the virtual machines running on top of it, trying to avoid problems that modern storage cluster have.
We will also show how different the performance can be comparing real and virtual machines, measuring access to computing, storage and network resources, trying to provide hints on server configuration where possible.
This work has been driven by the project called Worker Nodes on Demand Service (WNoDeS), developed by INFN, a framework designed to offer local, grid or cloud-based access to computing and storage resources, preserving maximum compatibility with existing computing center policies and work-flows.

The International Symposium on Grids and Clouds and the Open Grid Forum - ISGC2011,
March 25-30, 2011
Academia Sinica, Taipei, Taiwan

[*]Speaker.

1. Introduction

The INFN WNoDeS [1] (Worker Nodes on Demand Service) is a virtualization architecture targeted at Grid/Cloud integration. It provides transparent user interfaces for Grid, Cloud and local access to resources, re-using several existing and proven software components like Grid authentication and authorization mechanisms, KVM-based virtualization, local accounting, monitoring and work-flows, and data center schedulers. WNoDeS is in production at the INFN Tier-1 [2], Bologna, Italy since November 2009. Several million production jobs, including those submitted by experiments running at the LHC [3], have been processed by WNoDeS. Currently, about 2,000 VMs are dynamically created and managed within the INFN Tier-1 computing infrastructure. Recently, WNoDeS has been installed by an Italian WLCG Tier-2 site, with other domestic and international sites considering its adoption.

1.1 Key WNoDeS Characteristics

WNoDeS uses Linux KVM [4] to virtualize resources on-demand; the resources are available and customized for:

- direct job submissions by local users,
- Grid job submissions (with direct support for the EMI CREAM-CE and WMS components),
- instantiation of Cloud resources, instantiation of Virtual Interactive Pools (VIP)

VM (Virtual Machine) scheduling is handled by a LRMS (a "batch system software"): there is no need to develop special (and possibly unscalable, inefficient) resource brokering systems. The LRMS is totally invisible to users for e.g. Cloud instantiations; in particular, there is no concept of a "Cloud over Grid" or "Grid over Cloud" hierarchy: WNoDeS simply sees and uses all resources, dynamically presenting them to users as users want to see and access them.

2. Scaling locally distributed storage to thousands of VMs

The distributed file system adopted by the INFN Tier-1 is GPFS [5], serving about 8 PB of disk storage directly, and transparently interfacing to 10 PB of tape storage via the INFN GEMSS [6] system (an Mass Storage Solution based on StoRM/GPFS). An important scalability issue, which is not strictly GPFS-specific, is that any CPU core may become a GPFS (or any other distributed FS) client. This leads to GPFS clusters of several thousands of nodes (WNoDeS currently serves about 2,000 VMs at the INFN Tier-1). This is a large number, even according to the GPFS developers (IBM). Special care and tuning are therefore required, to mitigate the risk of impacting performance and functionality of the cluster. This situation will only get worse with the steady increase in the number of CPU cores in modern processors. With this problem in mind, we began to investigate two alternatives to mounting the GPFS FS on every VM, both assuming that an HV (HyperVisor) would distribute GPFS data to its own VMs, acting as a proxy. The alternatives considered are `sshfs` [7], a FUSE-based (Filesystem in USerspacE) solution, and a GPFS-to-NFS export (see figure 1).

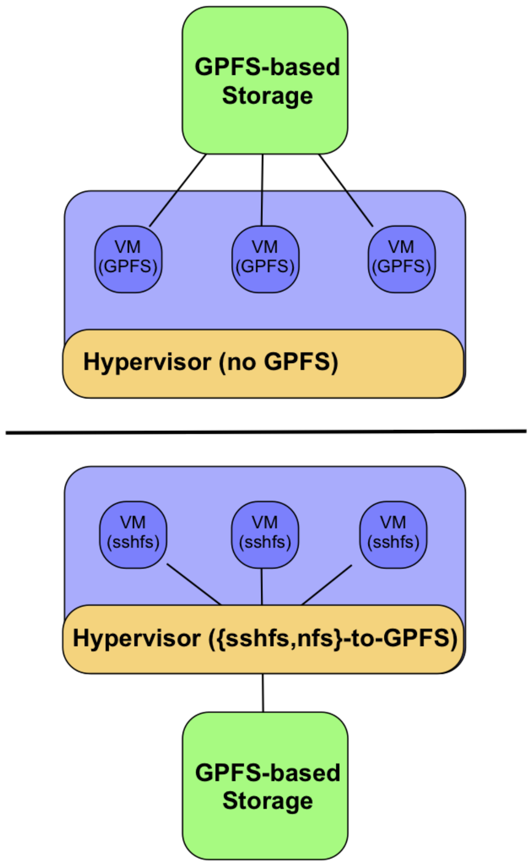

Figure 1: Alternatives to GPFS mounting on VMs
Top: the current solution. Bottom: a possible alternative.

2.1 SSHFS

`sshfs` allows to mount and use remote directory trees as if they were local. On the remote side, `sshfs` requires no special software except for a modern SSH server supporting the SFTP extension (Secure FTP). All modern Linux distributions support this extension, which was added to OpenSSH in version 2.3.0.

`sshfs` is built upon the FUSE user-space filesystem framework project. FUSE allows user-space software, SSH in this case, to present a virtual filesystem interface to the user. This is similar to how the `/proc` and `/sys` filesystems present kernel internals in the form of files in a directory tree. `sshfs` connects to the remote system and does all the necessary operations to provide the look and feel of a regular filesystem interface for remote files.

During our tests, `sshfs` throughput showed to be constrained by encryption, even using the lowest possible encryption level. A marked improvement, resulting in throughput better than NFS, was obtained by using `sshfs` with no encryption through `socat` [8], especially when some tuning was applied. The `socat` program is a command-line based utility that establishes two bidirectional

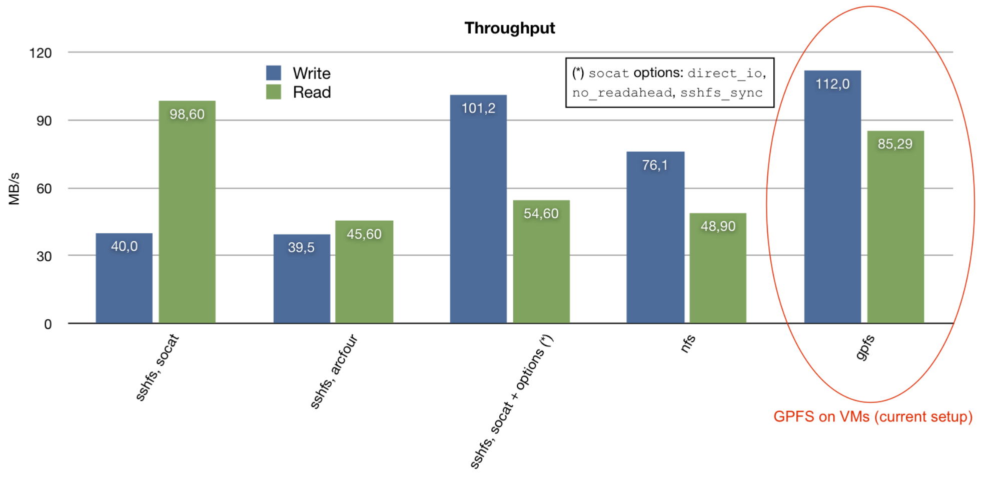

Figure 2: `sshfs` vs. `nfs`: throughput

byte streams and transfers data between them.

2.2 Results

Figure 2 compares the throughput performance of the current solution adopted by WNoDeS (GPFS on VMs) with the approaches described in this paragraph. The figure shows that `sshfs` using `socat` associated to suitable tuning options performs very well and could be seriously be taken into account as an alternative to the current approach. NFS is still a valid solution, although performances are not as good as with `sshfs`. The optimizations we have employed are `direct_io` (use direct I/O), `no_readahead` (do not perform speculative readahead and use synchronous reads instead) and `sshfs_sync` (use synchronous writes).

Figure 3 shows that, for what concerns cpu utilization, in some circumstances `sshfs` associated with `socat` performs even better than our current solution. Given these results, we can say that an alternative to the direct mount of GPFS filesystems on thousands of VMs may be available via hypervisor-based gateways, distributing data to the VMs local to the hypervisors. Both `nfs` and `sshfs` gateway perform with satisfactory results; the best solution seems to be a `sshfs`-based gateway, once no encryption is applied and some tuning parameters are set. However, special care needs to be taken when using `sshfs`-based gateways when dealing with user permissions. In particular, permissions on the client side must match those on the server. Anyway, if programs dynamically changing job permissions are used on the client side (e.g., `GLExec`-based programs [9]), a permission mismatch may occur. In this case one would need to either provision for managing these special cases, or employ a different solution. Support for `sshfs` and `nfs` gateways is scheduled to be included in WNoDeS version 2, whose release timeframe is currently Autumn 2011.

3. WNoDeS VM performance improvements

WNoDes uses KVM-based VMs, exploiting the KVM `-snapshot` flag. This allows us to

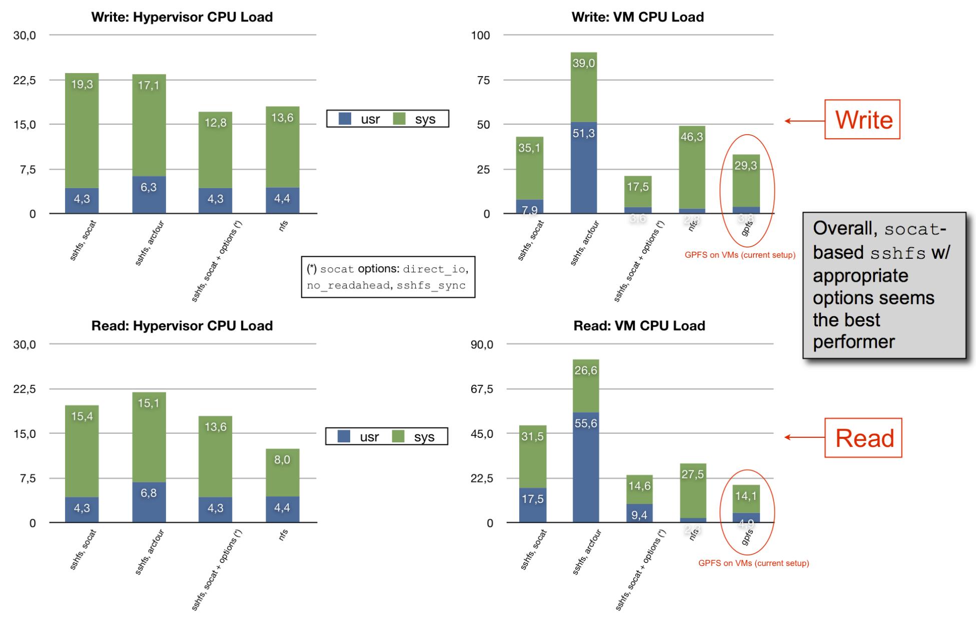

Figure 3: sshfs vs. nfs: CPU usage

download (via either `http` or Posix I/O) a single read-only VM image to each hypervisor, and run VMs writing automatically purged delta files only. This saves substantial disk space and time, which would be needed if one had to locally replicate multiple images. We do not run VMs stored on remote storage since, at the INFN Tier-1, the network layer is stressed out enough by user applications. We decided to perform several tests comparing the standard operating system used in WNoDeS hypervisors (SL5 [10]) with the latest available version. In all tests, since SL6 was not available at the time of testing, we used RHEL6 [11]. Our main goal was to understand latest improvements in technology (both hardware and software) and how to integrate them into next versions of WNoDeS.

To test the performance we measured:

- CPU, with HEP-Spec06 [13]
- disk I/O, with iozone [12]
- network I/O, with iperf [16]

Since `virtio-net` (the paravirtualized network driver optimized for KVM) was already been proven many times in the past to be efficient (90% or more of wire speed) we tested the Single Root I/O Virtualization (SR-IOV) specification [17], a new approach to network virtualization developed by the PCI-SIG (PCI Special Interest Group) and implemented by some vendors. The SR-IOV specification is a standard for a type of PCI-passthrough which natively shares a single device to multiple guests. SR-IOV reduces hypervisor involvement by specifying virtualization compatible memory spaces, interrupts and DMA streams and should improve I/O performance for virtualized guests. SR-IOV enables a Single Root Function (for example, a single Ethernet port) to appear as multiple, separate physical devices.

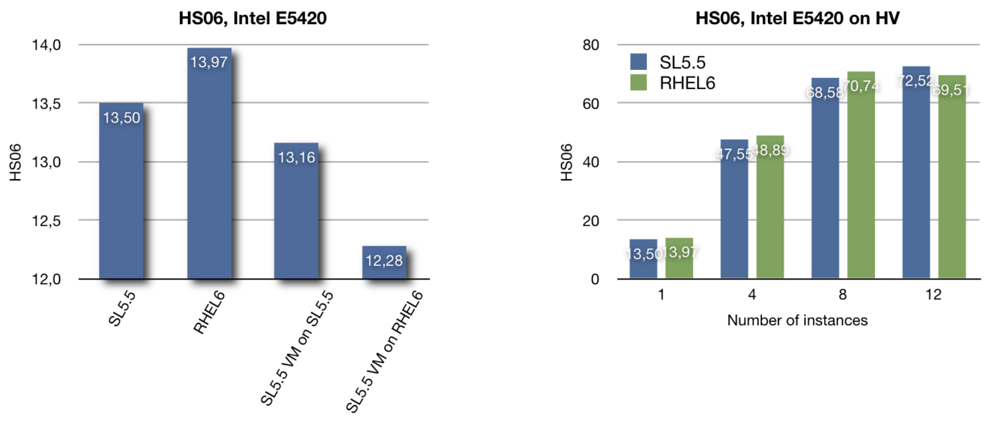

Figure 4: SL5 vs RHEL6

3.1 Testing environment

The hardware used for the test was composed of 1 DELL blade server with two Intel E5420 CPUs, 16GB RAM and two 10k SAS disks connected to a LSI Logic RAID controlled (disks configured with **RAID0** option). As for the OS installed:

- Host OS SL5: Scientific Linux (SL) 5.5 x86_64, kernel 2.6.18-194.32.1.el5, kvm-83-164.el5_5.9
- Host OS RHEL6:Red Hat Enterprise Linux 6.0 [11], kernel 2.6.32-71, qemu-kvm 0.12.1.2-2.113
- VM OS: Scientific Linux (SL) 5.5 x86_64, kernel 2.6.18-194.32.1.el5

The machine used for the SR-IOV test was slightly different, mounting two Intel E5520 CPUs, 24 GB RAM and an Intel **82576** SR-IOV enabled network adapter card. We tried to disable disk caching on VMs in all tests. The iozone [12] command line used was the following:

```
iozone -Mce -I -+r -r 256k -s <2xRAM>g -f <filepath> -i0 -i1 -i2
```

3.2 CPU performance

HEP-Spec [13] is the standard CPU performance benchmark test used in the High-Energy Physics (HEP) community, and is based on a subset of the SPEC benchmark [15]. We performed a wide number of tests in order to quantify the difference in performance for the two operating systems used as hypervisors, running the same VM image [14].

Figure 4 shows a slight performance increase of RHEL6 vs. SL5.5 on the hypervisor ($\sim$+3%, exception made for 12 instances, where performance decreases by $\sim$4%). The real surprising result was found measuring VM performance. In fact, the performance penalty of SL5.5 VMs on a SL5.5 hypervisor is $\sim$2.5%, and this may be acceptable. However, we measured an unexpected performance loss of SL5.5 VMs on a RHEL6 hypervisor (vs. the SL5.5 hypervisor) of $\sim$7%. Many reasons may lead to this result. We suspect that a key could be that RHEL6 is a rather new OS, subject to performance fixes and tuning; also, several new functions come into play on

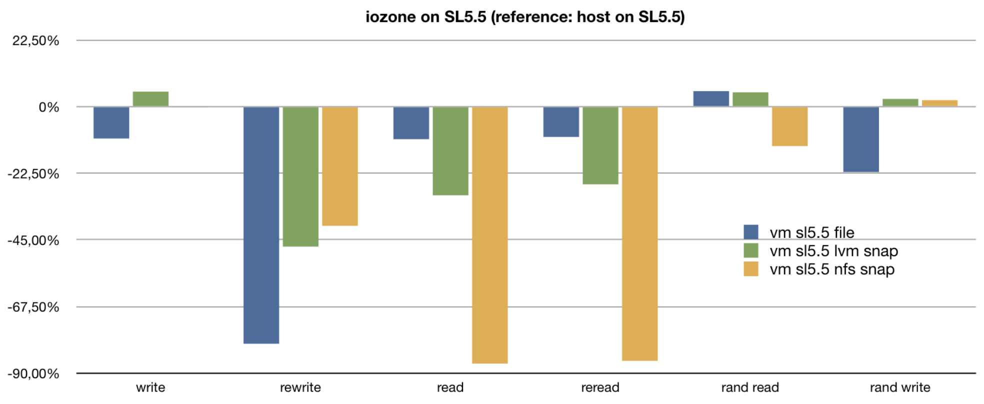

Figure 5: iozone on SL5.5 (SL5.5 VMs)

RHEL6: Kernel Samepage Merging (KSM) [18] may be important here, and it is possible that this service, that should be used to optimize memory utilization, may lead to performance loss. We will investigate this point further in future tests; at any rate, it is worth mentioning that the RHEL6 hypervisor has showed better stability under heavy load, compared to the SL5 one.

3.3 Disk performance

Disk access speed is critical for every virtualization technology and our tests showed that a lot is still to be improved. We performed three different measurements, using different approaches for disk I/O:

- the filesystem tested was a file on the host machine.
- the filesystem tested was an LVM (Logical Volume Manager) partition on the hypervisor, exported exclusively to the VM. The KVM `-snapshot` flag was used.
- the filesystem tested was an NFS mount, exported by the hypervisor. The KVM `-snapshot` flag was used.

Figure 5 shows the performance of a VM compared to its baseline (the physical machine), both running on SL5.5. Six measures have been taken: sequential read and write performance, random read and write performance, and finally reread and rewrite performance. All these test are significant to understand the level of the disk emulation and all have been taken into account. We can clearly understand that disk access is still very problematic: all the approaches show poor performance when compared to the host. We can anyway summarize that an approach based on an LVM partition is the best, generally providing better throughput than the others.

Figure 6 shows the performance of a VM compared to a RHEL6 baseline. Consistently with what was seen with some CPU performance tests, `iozone` on RHEL6 surprisingly performs often worse than on SL5.5. RHEL6 supports native Kernel Asynchronous I/O (AIO) and `preadv/pwritev` to group together memory areas before reading or writing them. This is possibly the reason for some funny results (namely, unbelievably good performance) of the `iozone` benchmark.

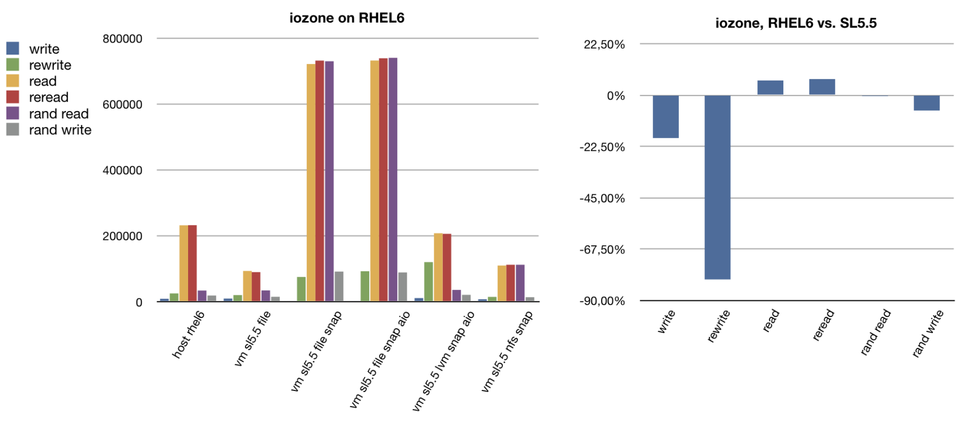

Figure 6: iozone on RHEL6 (SL5.5 VMs)

As a consequence of this set of tests, we do not plan to upgrade our hypervisors to RHEL6/SL6 until we are able to get reasonable results and improvements in this area.

Based on these tests, the next WNoDeS release will still use the `-snapshot` flag, but performance will be significantly improved compared to the current release by letting `-snapshot` operate on the root partition only. The `/tmp` partition and local user data will instead reside on a host-based LVM partition, which will be dynamically created at VM instantiation time.

3.4 Network performance

Network performance is an essential parameter to measure: with `iperf` [16] we tested throughput both inbound and outbound directions. The options we used are `-w256k -P 5 -t 900`. This implies a TCP window size of 256k, 5 parallel connections and a duration of 900 seconds (15 minutes). Network performance is much more comforting than the previous disk tests. Indeed with current technology it is possible to easily reach wire speed, without any specific tweaking or tuning.

The network-related measures we performed were just a cross-check to verify if the latest KVM implementation in RHEL6 brought about any discrepancy. Figure 7 shows that input is generally better than output, and no significant difference is present between a real and a virtual machine. Considering network speed not be a problem, we then decided to investigate performance of a relatively new SR-IOV device.

3.5 SR-IOV

A physical device with SR-IOV [17] capabilities can be configured to appear in the PCI configuration space as multiple functions. Each device has its own configuration space complete with Base Address Registers (BARs). This new “hardware” approach to network virtualization sounds promising but our tests did not show significant performance benefits; on the contrary, they showed performance to be in some cases worse than with simple `virtio-net`. Figure 7 shows that, even if throughput is comparable to the one found with host and `virtio`, SR-IOV CPU utilization and latency are both outperformed by `virtio-net`. This is rather disappointing, since this approach should take benefit of hardware acceleration.

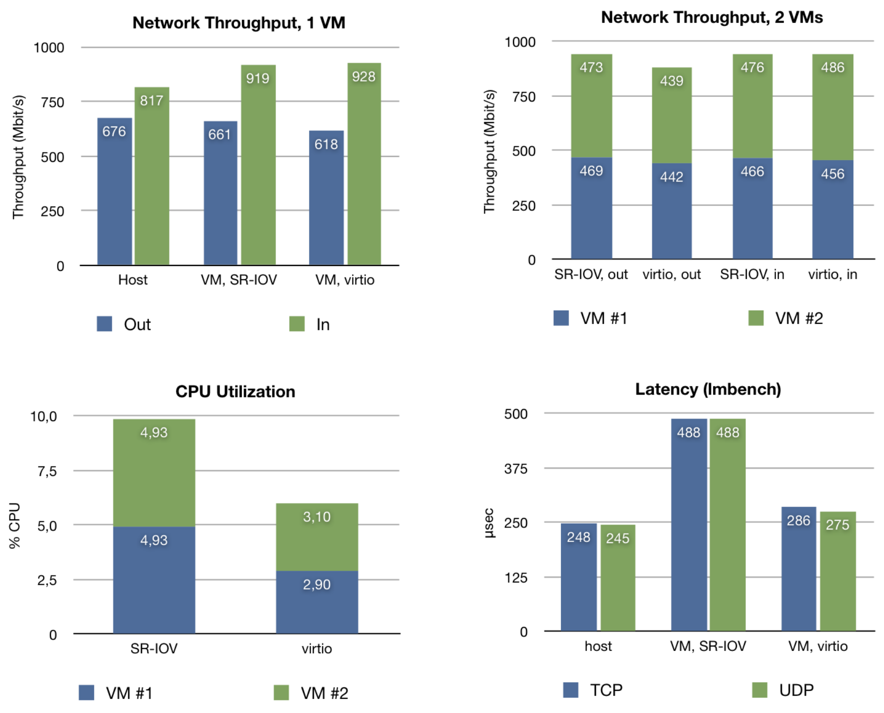

Figure 7: Network performance

SR-IOV should stand out in environments where more than a single gigabit connection is provided. However, in our tests we were only able to test a single gigabit connection, due to the limitation of the network adapter used. Tests on 10 Gbps cards are ongoing, and there we expect to see significant improvements, especially in terms of latency.

We can conclude that even if looking promising, right now SR-IOV is not a good solution for improving network performance of WNoDeS: the best solution is still virtio.

4. Conclusions

VM performance tuning requires detailed knowledge of system internals and sometimes of applications behavior. Many improvements of various types have generally been implemented in hypervisors and in VM management systems. Some not described here are:

- KSM (Kernel Samepage Merging) [18] to overcommit memory: due to the nature of our typical applications, we normally do not need to overcommit memory.
- VM pinning: the benefit of such a technique is debatable, since one has to watch out for I/O details and subtleties in CPU hardware architectures.
- Advanced VM brokerage. WNoDeS fully uses LRMS-based brokering for VM allocations; thanks to this, algorithms for grouping VMs to, for example, partition I/O traffic (to group together all VMs belonging to a certain VO/user group, for instance) or to minimize the number of active physical hardware (to suspend / hibernate / turn off unused hardware, for

instance) can be easily implemented. Whether to do it or not actually substantially depends on the data centers infrastructure and applications.

WNoDeS is facilitated in this type of performance tuning by the fact that it only focuses on Linux KVM as an hypervisor; there is no intention to make it more general and support other hypervisors. The steady increase in the number of cores per physical hardware has a significant impact in the number of virtualized systems even on a medium-sized farm. This is important both for access to distributed storage, and for the set-up of traditional batch system clusters (e.g. the size of a batch farm easily increases by an order of magnitude with VMs).

Finally, we believe that the main issue with virtualization is not so much in virtualizing resources, not even a large number of them. Troubles are potentially much more to be found in the design, implementation and operation of a dynamic, scalable, extensible, efficient architecture, which should be integrated with local, Grid, Cloud access interfaces and with large storage systems. WNoDeS was developed with these considerations in mind, and we think that it can be a possible solution to these requirements.

References

[1] WNoDeS Website: http://web.infn.it/wnodes

[2] INFN-CNAF Website: http://www.cnaf.infn.it

[3] LHC Website: http://lhc.web.cern.ch/lhc

[4] KVM Website: http://www.linux-kvm.org

[5] General Parallel File System Website: http://www-03.ibm.com/systems/software/gpfs/

[6] A. Carbone et al., *A Novel Approach for Mass Storage Data Custodial*, [2008 IEEE Nuclear Science Symposium Conference Record - N70-4]

[7] SSHFS on Wikipedia: http://en.wikipedia.org/wiki/SSHFS

[8] Socat Website: http://freshmeat.net/projects/socat

[9] GLExec Website: https://www.nikhef.nl/pub/projects/grid/gridwiki/index.php/GLExec

[10] Scientific Linux Website: https://www.scientificlinux.org/

[11] Red Hat Enterprise Linux Website: http://www.redhat.com/rhel/

[12] IOzone Website: http://www.iozone.org

[13] HEP-SPEC06 Website: https://twiki.cern.ch/twiki/bin/view/FIOgroup/TsiBenchHEPSPEC

[14] A. Chierici et al., *A quantitative comparison between Xen and KVM* [2009 Journal of Physics: Conference series]

[15] SPEC Website: http://www.spec.org/

[16] Iperf Website: http://sourceforge.net/projects/iperf/

[17] SR-IOV Primer: http://download.intel.com/design/network/applnots/321211.pdf

[18] Kernel Samepage Merging: http://www.linux-kvm.org/page/KSM