Performance Improvements in a Large-Scale Virtualization System

Davide Salomoni, Anna Karen Calabrese Melcarne,
Gianni Dalla Torre, Alessandro Italiano,
Andrea Chierici

Workshop CCR-INFNGrid, 2011

Why this presentation

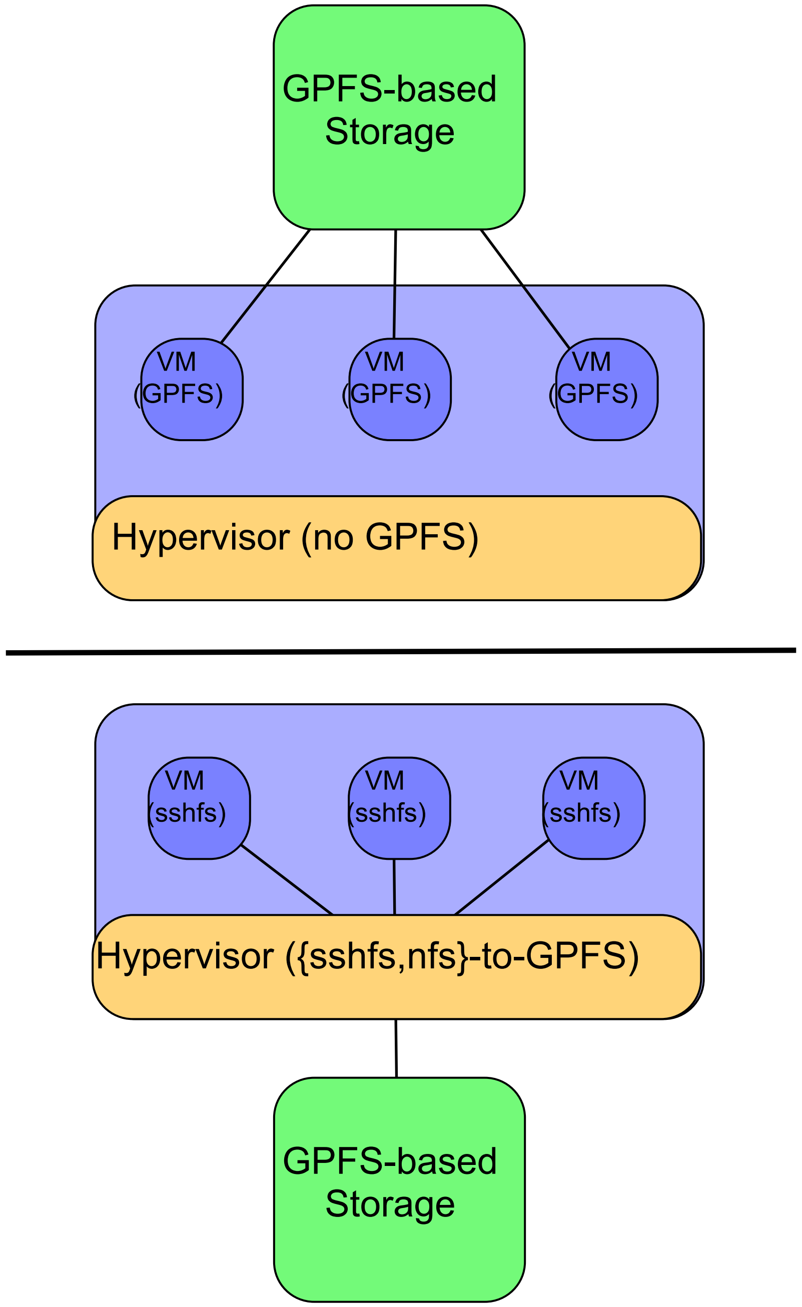
- CNAF deeply involved in virtualization
  - WNoDeS
  - CCR Virtualization group
  - Modern CPU “ask” to be used with virtualization
- Will show all the tests we performed aimed to solve bottlenecks and to improve virtual machines speed
- These results do not apply only to WNoDeS
- See also SR-IOV poster

WNoDeS Release Schedule

- WNoDeS 1 released in May 2010
- WNoDeS 2 “Harvest” public release scheduled for September 2011
  - More flexibility in VLAN usage - supports VLAN confinement to certain hypervisors only
  - `libvirt` now used to manage and monitor VMs
    - Either locally or via a Web app
  - Improved handling of VM images
    - Automatic purge of “old” VM images on hypervisors
    - Image tagging now supported
    - Download of VM images to hypervisors via either `http` or Posix I/O
  - Hooks for porting WNoDeS to LRMS other than LSF
  - Internal changes
    - Improved handling of Cloud resources
    - New plug-in architecture
  - Performance, management and usability improvements
    - Direct support for LVM partitioning, significant performance increase with local I/O
    - Support for local `sshfs` or `nfs` gateways to a large distributed file system
    - New web application for Cloud provisioning and monitoring, improved command line tools

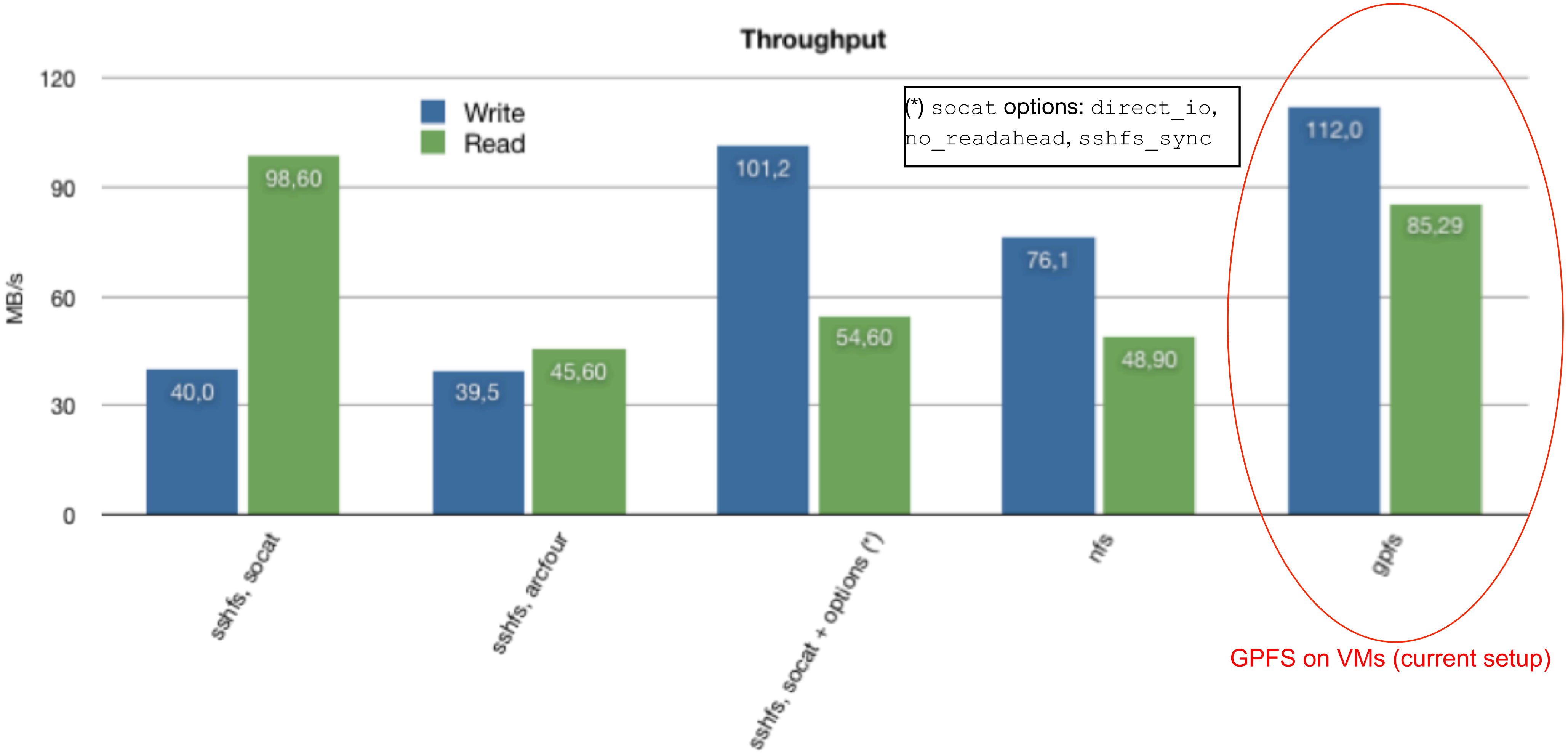
Alternatives to mounting GPFS on VMs

- Preliminary remark: the distributed file system adopted by the INFN Tier-1 is GPFS
  - Serving about 8 PB of disk storage directly, and transparently interfacing to 10 PB of tape storage via INFN's GEMSS (an MSS solution based on StoRM/ GPFS)
- The issue, 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 large, even according to IBM, requires special care and tuning, and may impact performance and functionality of the cluster
  - This will only get worse with the steady increase in the number of CPU cores in processors
  - We investigated two alternatives, both assuming that an HV would distributed data to its own VMs
    - `sshfs`, a FUSE-based solution
    - a GPFS-to-NFS export



`sshfs` vs. `nfs`: throughput

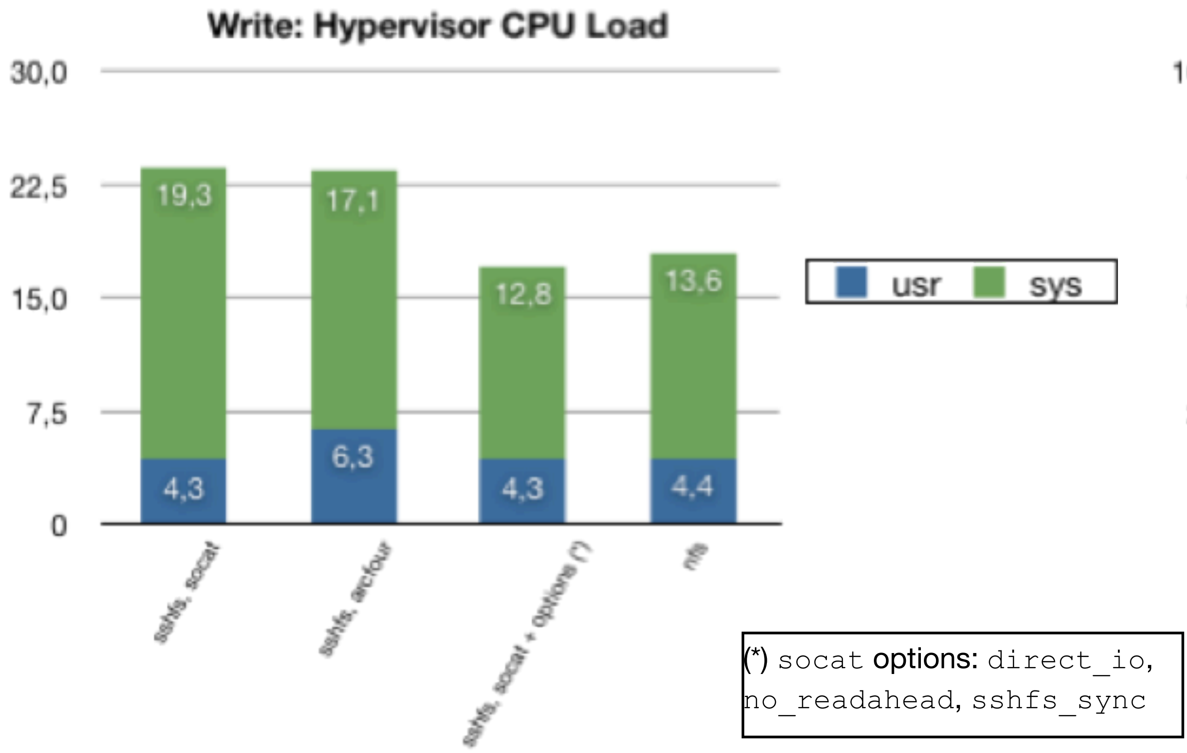
- `sshfs` throughput constrained by encryption (even with the lowest possible encryption level)
- Marked improvement (throughput better than `nfs`) using `sshfs` with no encryption through `socat`, esp. with some tuning
  - □ File permissions are not straightforward with `socat`, though

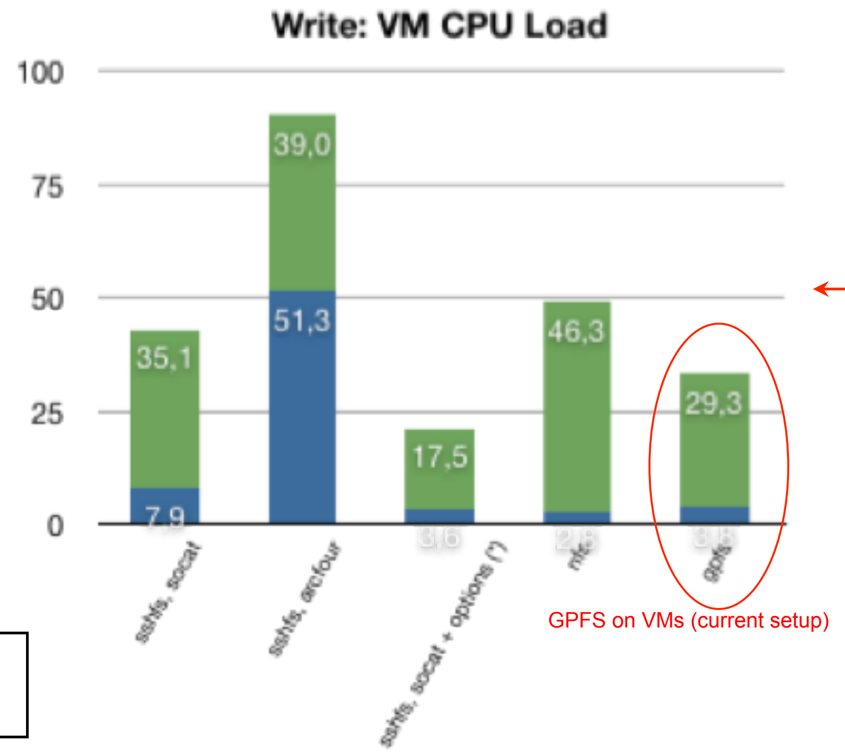


(*) `socat` options: `direct_io`, `no_readahead`, `sshfs_sync`

GPFS on VMs (current setup)

`sshfs` vs. `nfs`: CPU usage



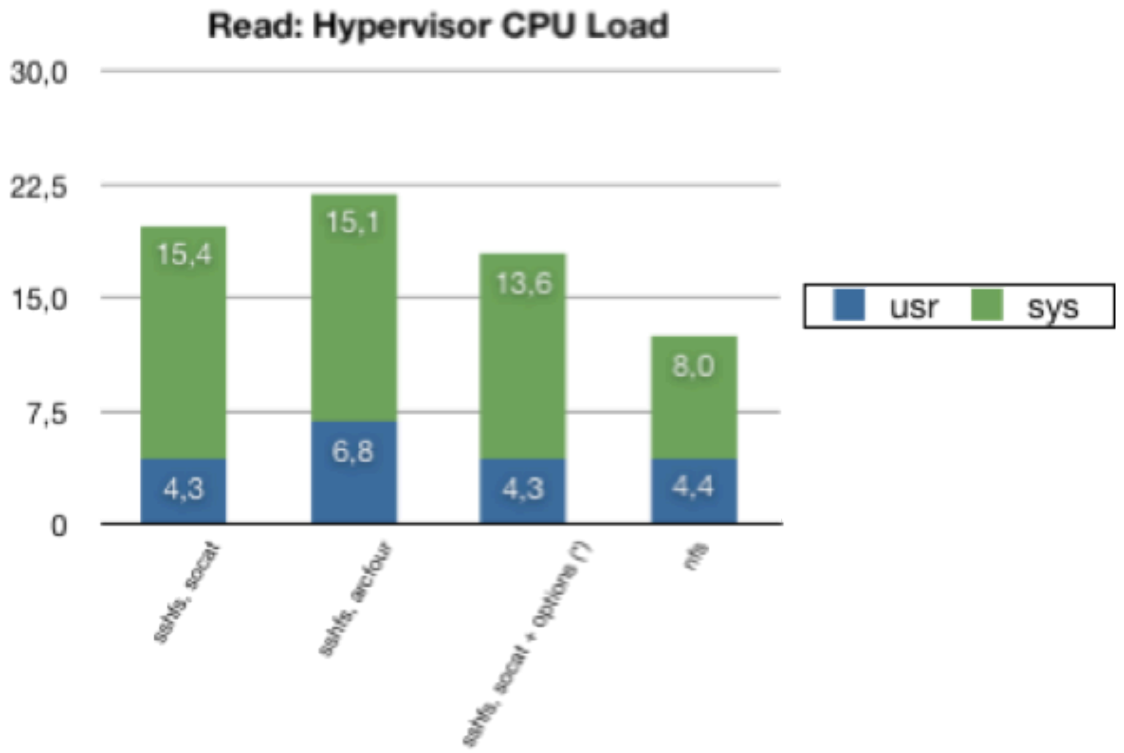


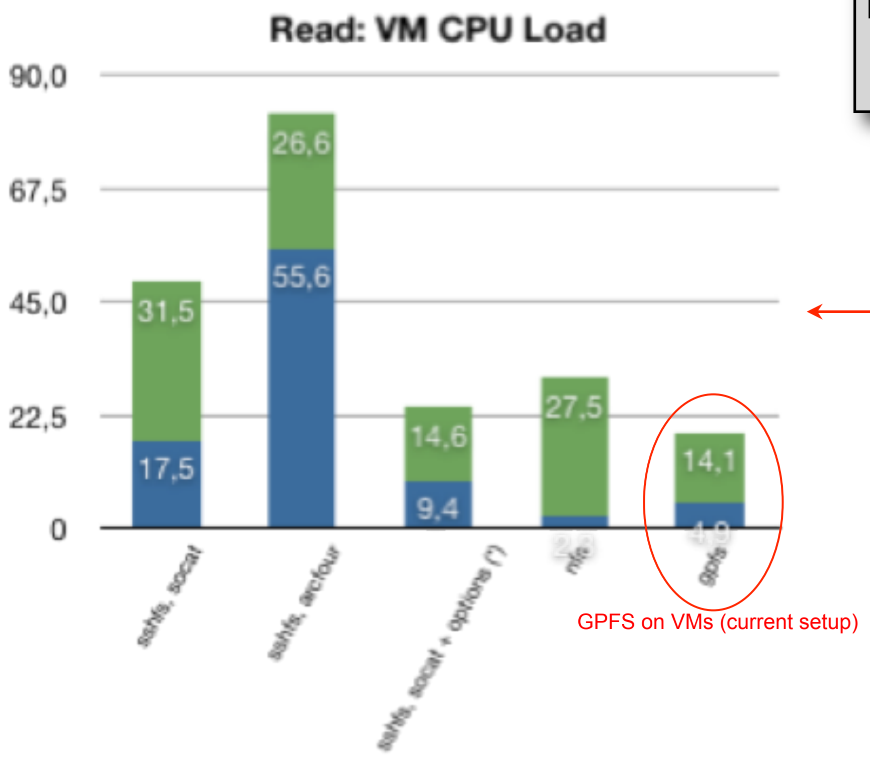
GPFS on VMs (current setup)

Write

(*) `socat` options: `direct_io`, `no_readahead`, `sshfs_sync`

Overall, `socat`-based `sshfs` w/ appropriate options seems the best performer





GPFS on VMs (current setup)

Read

`sshfs` vs. `nfs` Conclusions

- An alternative to direct mount of GPFS filesystems on thousands of VMs is available via hypervisor-based gateways, distributing data to VMs
- Overhead, due to the additional layer in between, is present. Still, with some tuning it is possible to get quite respectable performance
  - `sshfs`, in particular, performs very well, once you take encryption out. But one needs to be careful with file permission mapping between `sshfs` and GPFS,
- Watch for VM-specific caveats
  - For example, WNoDeS supports hypervisors and VMs to be put in multiple VLANs (VMs themselves may reside in different VLANs)
- Support for `sshfs` or `nfs` gateways is scheduled to be included in WNoDeS 2 “Harvest”

- VirtFS (Plan 9 folder sharing over Virtio - I/O virtualization framework) investigation in the future, but native support by RH/SL currently missing

VM-related Performance Tests

- Preliminary remark: WNoDes uses KVM-based VMs, exploiting the KVM `-snapshot` flag
  - This allows us to 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 to locally replicate the images
  - We do not run VMs stored on remote storage - at the INFN Tier-1, the network layer is stressed out enough by user applications
- Tests performed:
  - SL6 vs SL5
    - Classic HEP-Spec06 for CPU performance
    - `Iozone` for local I/O
  - Network I/O:
    - `virtio-net` has been proven to be quite efficient (90% or more of wire speed)
    - We tested SR-IOV, see the dedicated poster (if you like, vote it! ☺)
  - Disk caching is (should have been) disabled in all tests
- Local I/O has typically been a problem for VMs
  - WNoDeS not an exception, esp. due to its use of the KVM `-snapshot` flag
  - The next WNoDeS release will still use `-snapshot`, but for the root partition only; `/tmp` and local user data will reside on a (host-based) LVM partition

Testing set-up

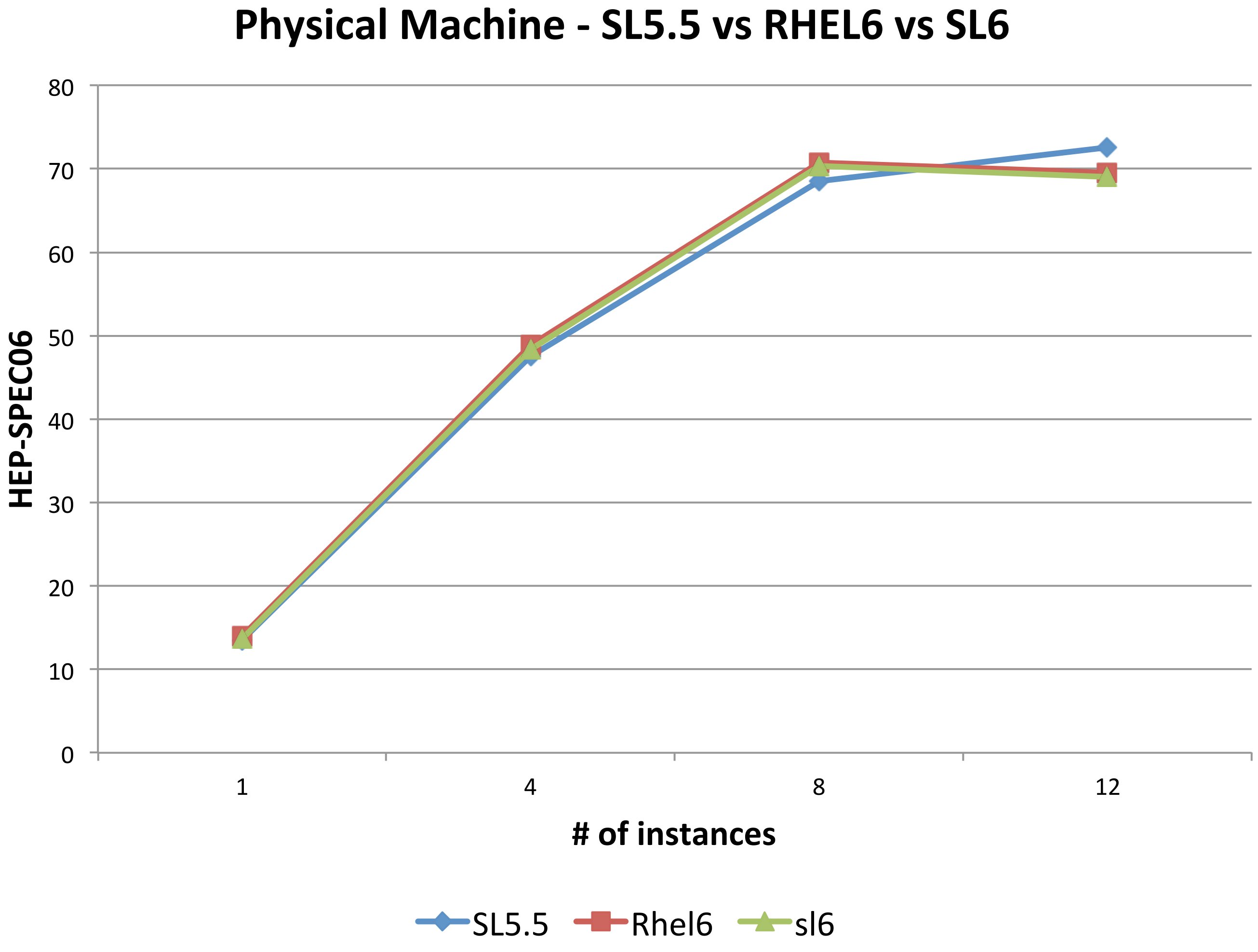
- HW: 4x Intel E5420, 16 GB RAM, 2x 10k rpm SAS disk using a LSI Logic RAID controller
- SL5.5: kernel 2.6.18-194.32.1.el5, kvm-83-164.el5_5.9
- SL 6: kernel 2.6.32-71.24.1, qemu-kvm-0.12.1.2-2.113
- SR-IOV: tests on a 2x Intel E5520, 24 GB RAM with an Intel 82576 SR-IOV card

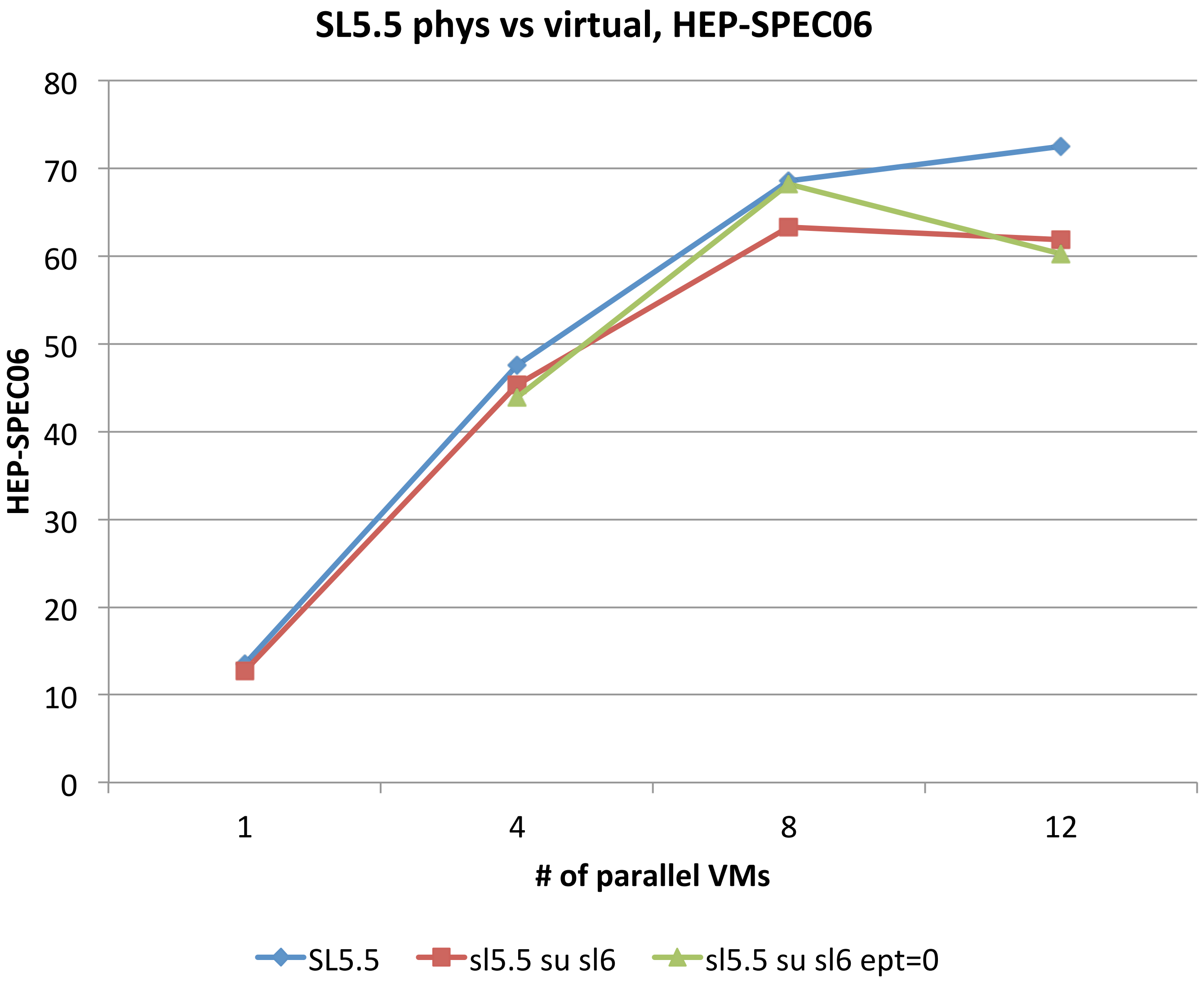
- iozone:

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

HS06 on Hypervisors and VMs (E5420)

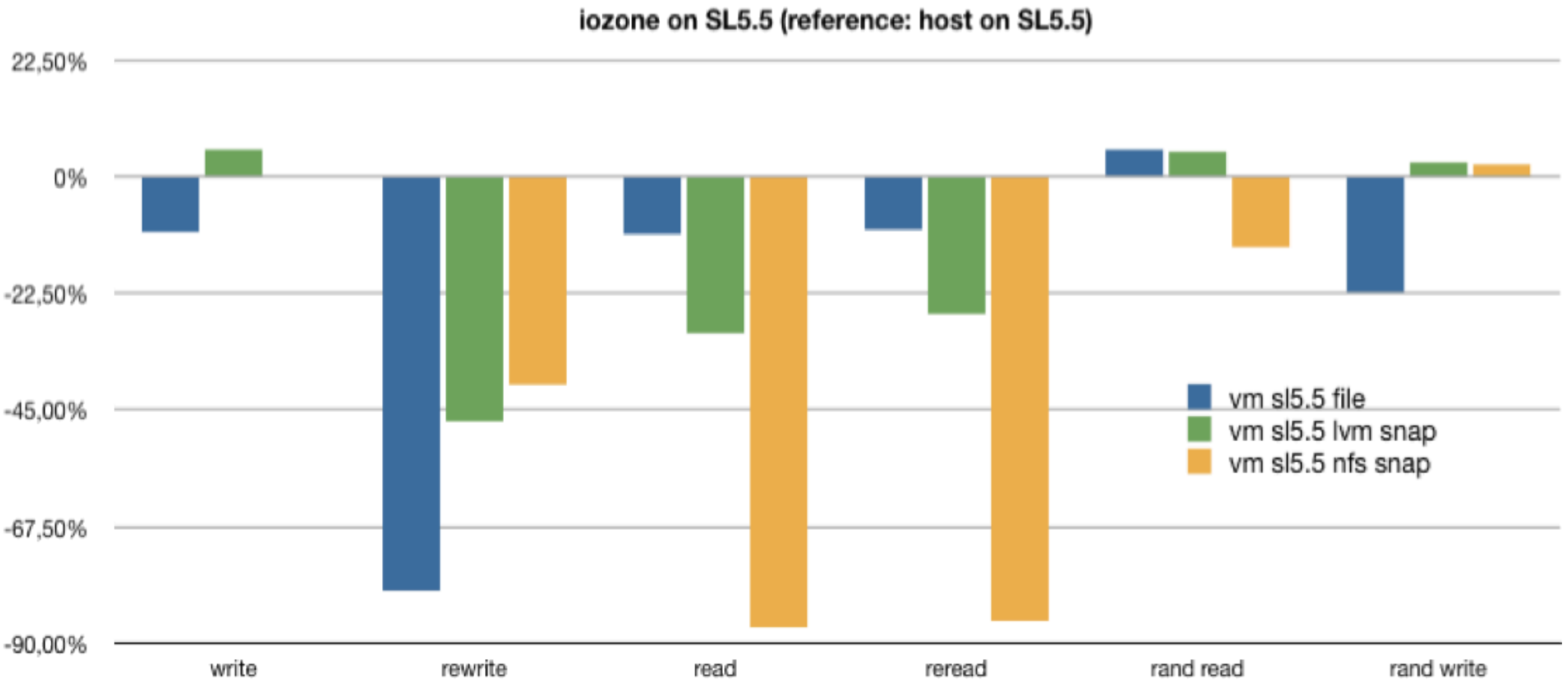
- Slight performance increase of SL6 vs. SL5.5 on the hypervisor
  - Around +3% (exception made for 12 instances: -4%)
- Performance penalty of SL5.5 VMs on SL5.5 HV: -2.5%
- Unexpected performance loss of SL5.5 VMs on SL6 vs. SL5.5 HV
  - ept — Extended Page Tables, an Intel feature to make emulation of guest page tables faster.





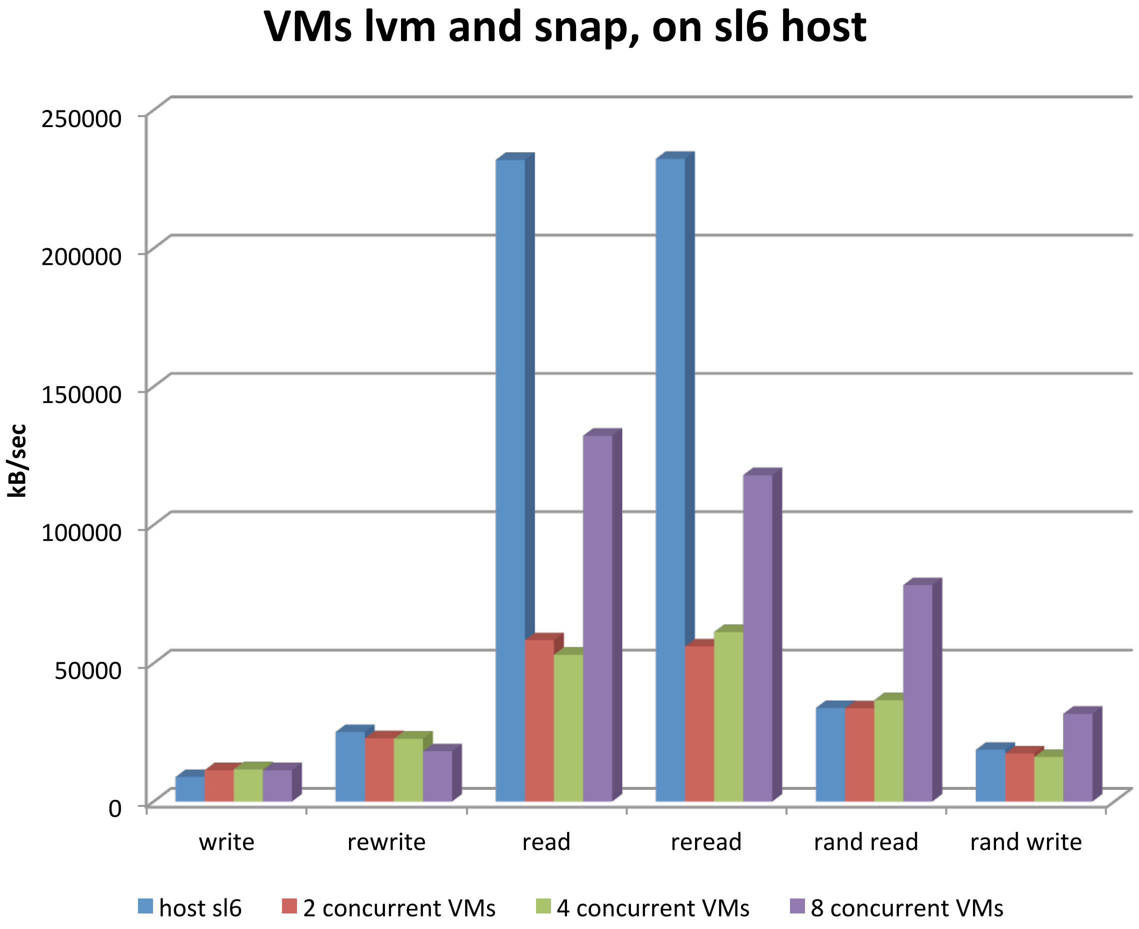
`iozone` on SL5.5 (SL5.5 VMs)

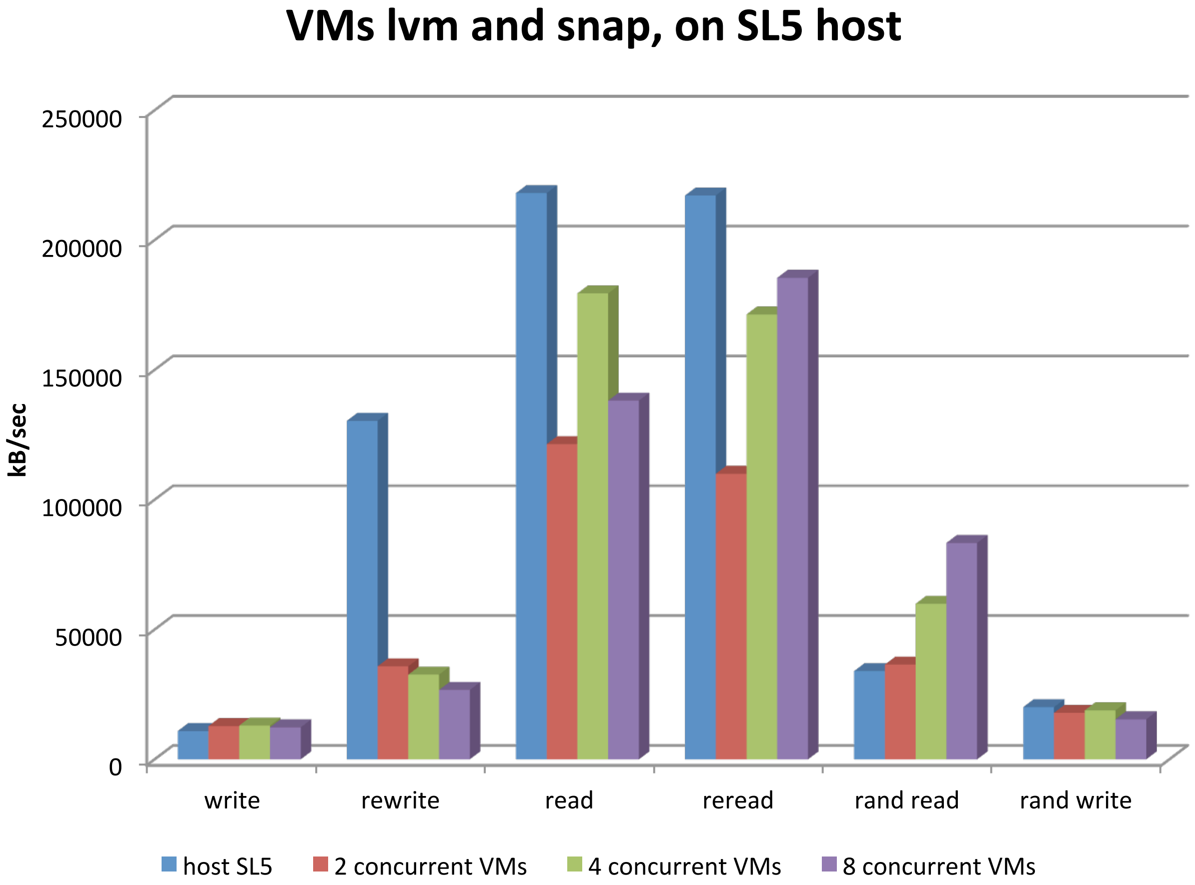
- `iozone` tests with caching disabled, file size 4 GB on VMs with 2GB RAM
- host with SL5.5 taken as reference
- VM on SL5.5 with just `-snapshot` crashed
- Based on these tests, WNoDeS will support `-snapshot` for the root partition and a (dynamically created) native LVM partition for `/tmp` and for user data
  - A per-VM single file or partition would generally perform better, but then we'd practically lose VM instantiation dynamism



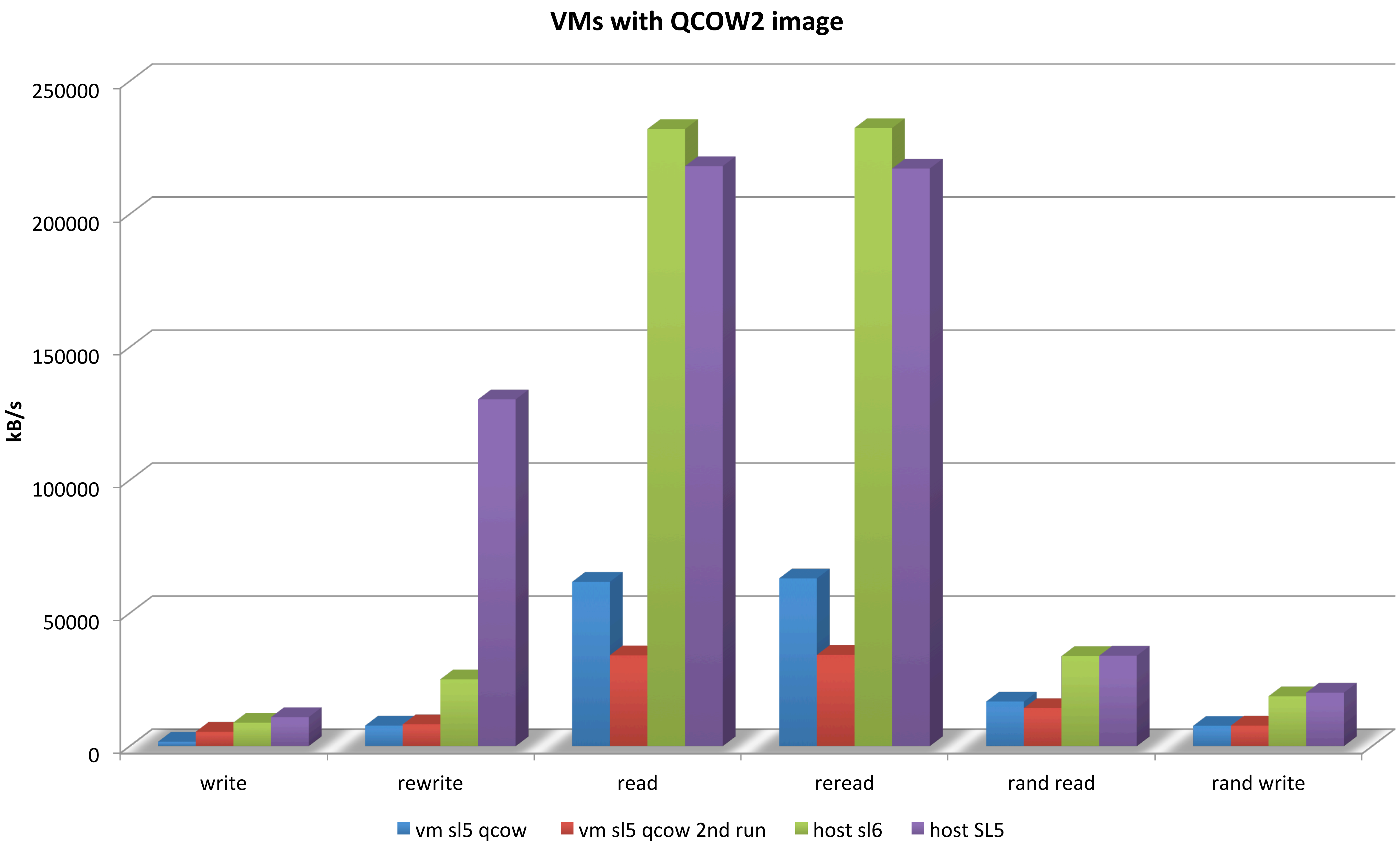
`iozone` on SL6 (SL5.5 VMs)

- Consistently with what was seen with some CPU performance tests, `iozone` on SL6 surprisingly performs often worse than on SL5.5
- Assuming RHEL6 performance will be improved by RH, using VM with `-snapshot` for the root partition and a native LVM patition for `/tmp` and user data in WNoDes seems a good choice here as well
  - □ But we will not upgrade HVs to SL6 until we are able to get reasonable results in this area



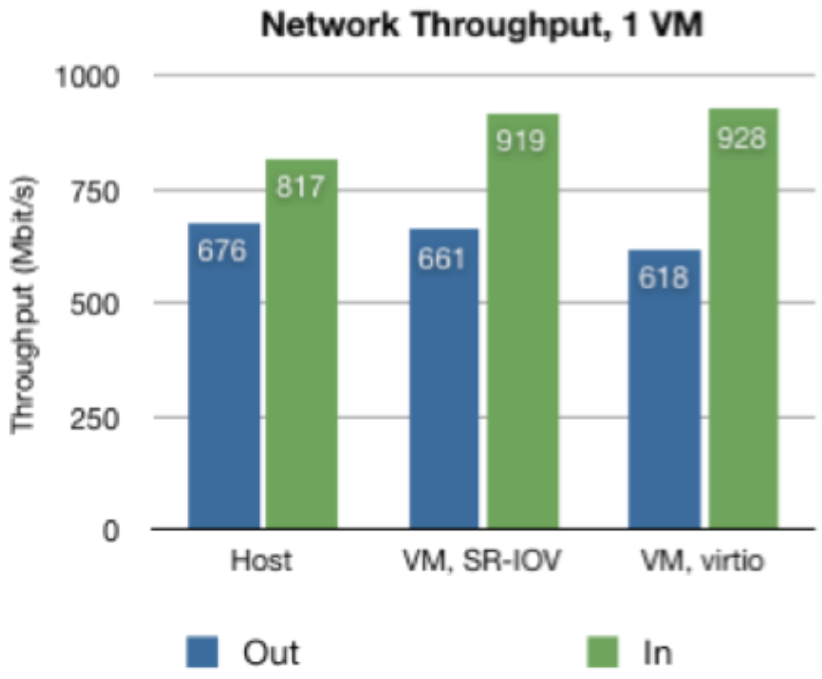


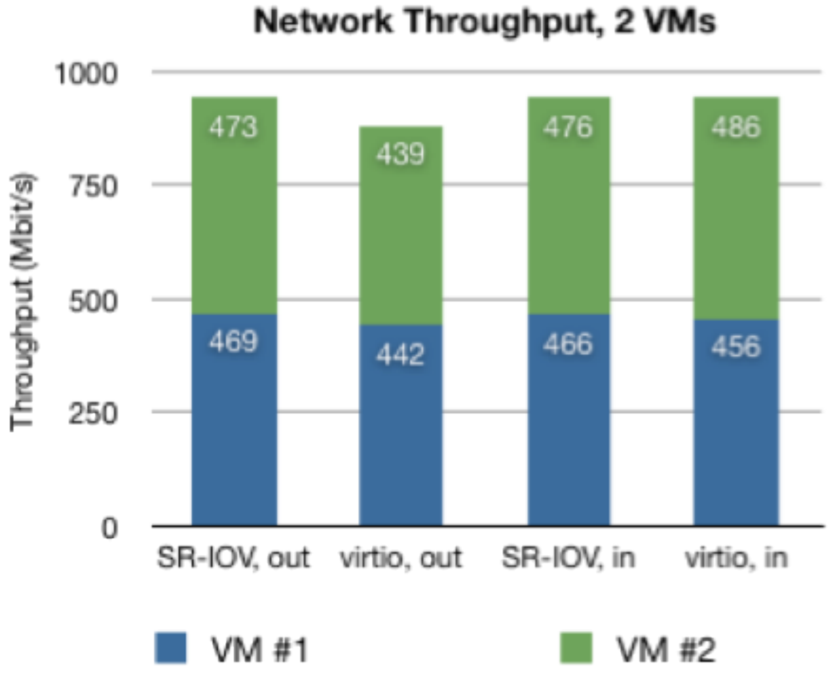
`iozone` on QCOW2 image file

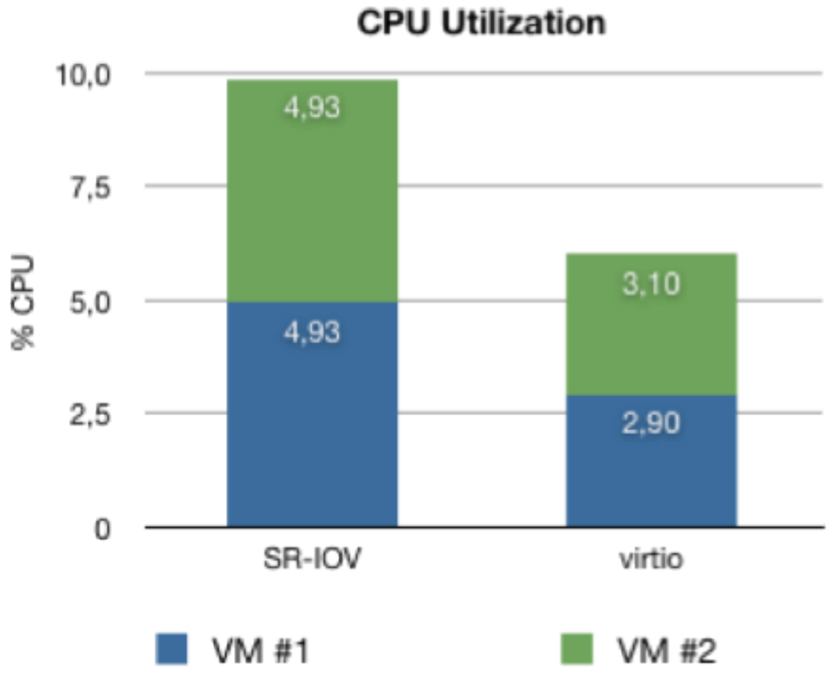


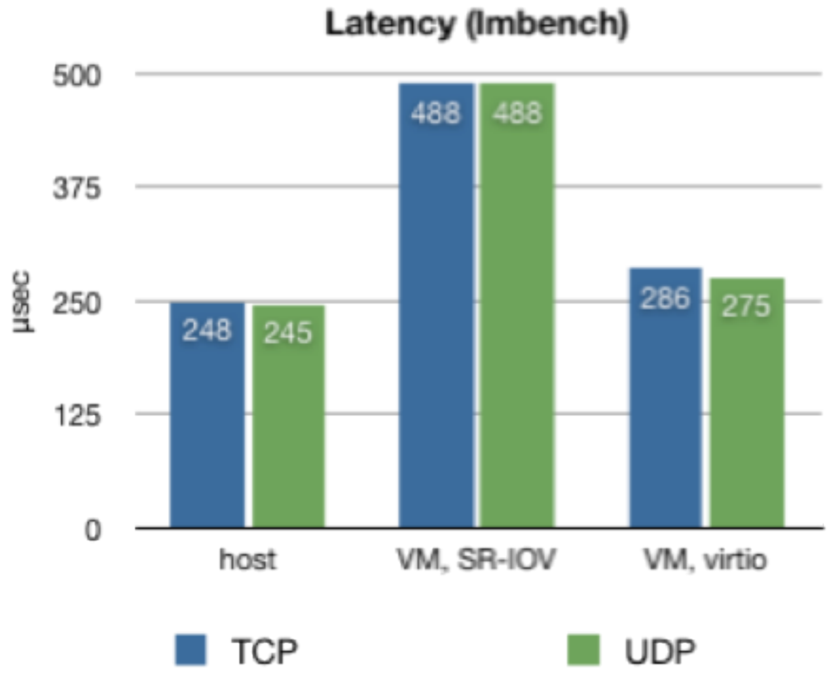
Network

- SR-IOV slightly better than virtio wrt throughput
- Disappointing SR-IOV performance wrt latency, CPU utilization


Network Throughput, 1 VM


Network Throughput, 2 VMs


CPU Utilization


Latency (lmbench)

The problem we see for the future

- Number of cores in modern CPUs is constantly increasing
- Virtualizing to optimize (cpu/ram) resources is not enough
  - O(20) cores per cpu will require 10GBps nics (at least at T1)
  - Disk i/o is still a problem (it was the same last year, no significant improvement has been done)

Technology improvements

- SSDs may help
  - Did not arrive on time to be tested ☹
  - Great expectations, but price will prevent massive adoption at least in 2011
- SR-IOV nics are very interesting
  - Drivers have to improve
- SL6: virtualization embedded
  - KSM, hugetlbfs, pci-passthrough
  - Still problems with performance
- KVM VirtFS: para-virtualized FS

Conclusions

- VM performance tuning still requires detailed knowledge of system internals and sometimes of application behaviors
  - Many improvements of various types have generally been implemented in hypervisors and in VM management systems. Some not described here are:
    - VM pinning. Watch out for I/O subtleties in CPU hardware architectures.
    - Advanced VM brokerage. WNoDeS fully uses LRMS-based brokering for VM allocations; thanks to this, algorithms for e.g. grouping VMs to partition I/O traffic (for example, to group together all VMs belonging to a certain VO/user group) or to minimize the number of active physical hardware (for example, to suspend / hibernate / turn off unused hardware) can be easily implemented (whether to do it or not depends much on the data centers infrastructure / applications)
- 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).
- The difficulty is not so much in virtualizing (even a large number of) resources. It is much more in having a dynamic, scalable, extensible, efficient architecture, integrated with local, Grid, Cloud access interfaces and with large storage systems.