Performance Improvements in a Large-Scale Virtualization System

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

INFN-CNAF, Bologna, Italy

ISGC 2011 - Taipei, 19-25 March, 2011

Outline

- Introduction to WNoDeS
- Scaling locally distributed storage to thousands of VMs
- WNoDeS VM performance improvements
- Conclusions

Introduction to WNoDeS

- The INFN WNoDeS (Worker Nodes on Demand Service) is a virtualization architecture targeted at Grid/Cloud integration
  - Providing transparent user interfaces for Grid, Cloud and local access to resources
  - Re-using several existing and proven software components, e.g. Grid AuthN/AuthZ, KVM-based virtualization, local workflows, data center schedulers
  - See http://web.infn.it/wnodes for details
- In production at the INFN Tier-1, Bologna, Italy since November 2009
  - Several million production jobs processed by WNoDeS (including those submitted by experiments running at the LHC)
  - Currently, about 2,000 dynamically created VMs
  - Integration with the INFN Tier-1 storage system (8 PB of disk, 10 PB of tape storage)
  - Also running at an Italian WLCG Tier-2 site, with other sites considering its adoption

Key WNoDeS Characteristics

- Uses Linux KVM 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)
    - See e.g. the WNoDeS talk on VIP at CHEP 2010, October 2010
- VM scheduling is handled by a LRMS (a “batch system software”)
  - No need to develop special (and possibly unscalable, inefficient) resource brokering systems
  - The LRMS is totally invisible to users for e.g. Cloud instantiations
- No concept of “Cloud over Grid” or “Grid over Cloud”
  - WNoDeS simply uses *all* resources and dynamically presents them to users as users want to see and access them
- At this conference, see also:
  - Grids and Clouds Integration and Interoperability: an Overview
  - A Web-based Portal to Access and Manage WNoDeS Virtualized Cloud Resources

WNoDeS Release Schedule

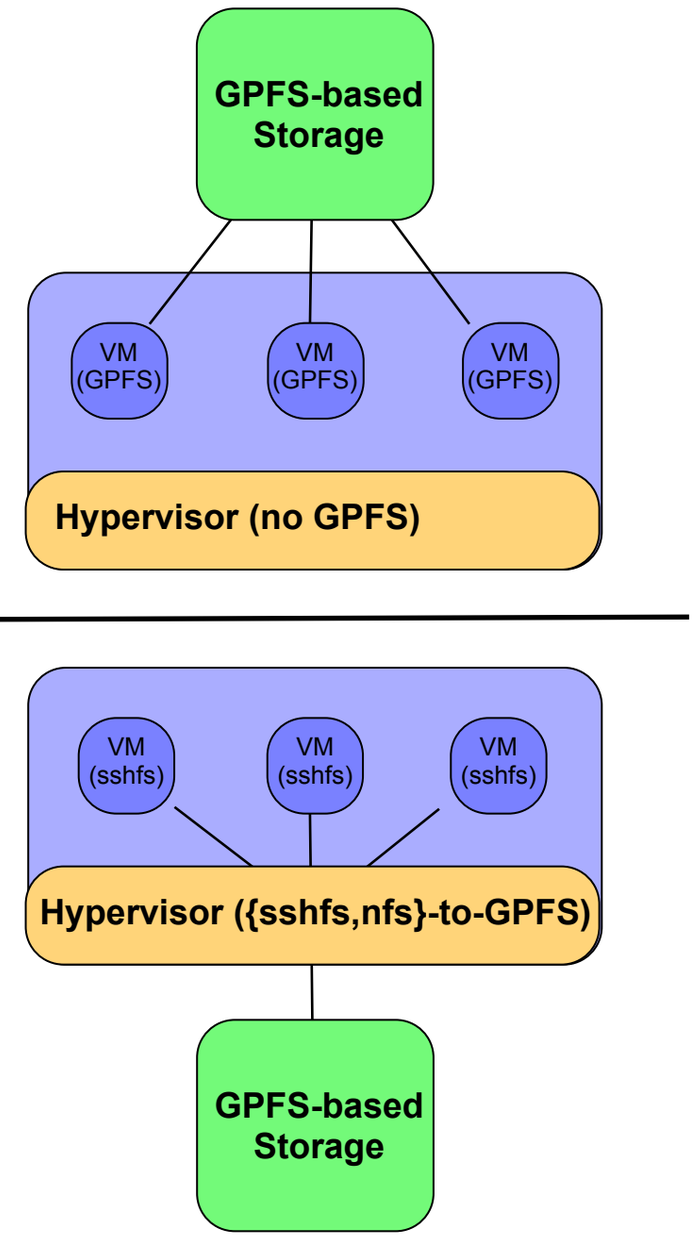
- WNoDeS 1 released in May 2010
- WNoDeS 2 “Harvest” public release scheduled for September 2011
  - More flexibility in VLAN usage - supports VLAN confinement to certain hypervisors only
    - Used at CNAF to implement a “Tier-3” infrastructure alongside the main Tier-1
  - `libvirt` now used to manage 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 Platform 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

Outline

- Introduction to WNoDeS
- Scaling locally distributed storage to thousands of VMs
- WNoDeS VM performance improvements
- Conclusions

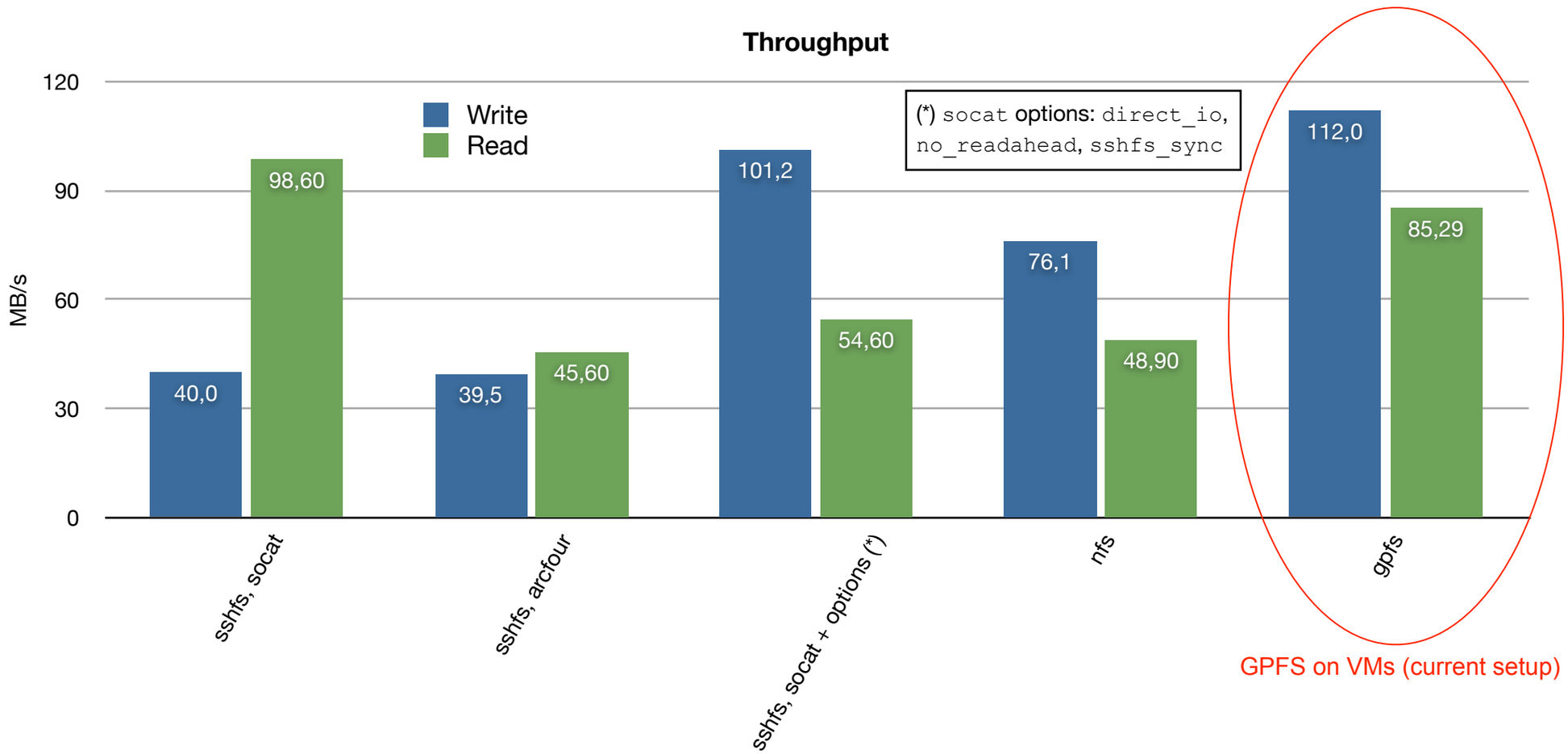
Alternatives to mounting GPFS on VMs

- 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

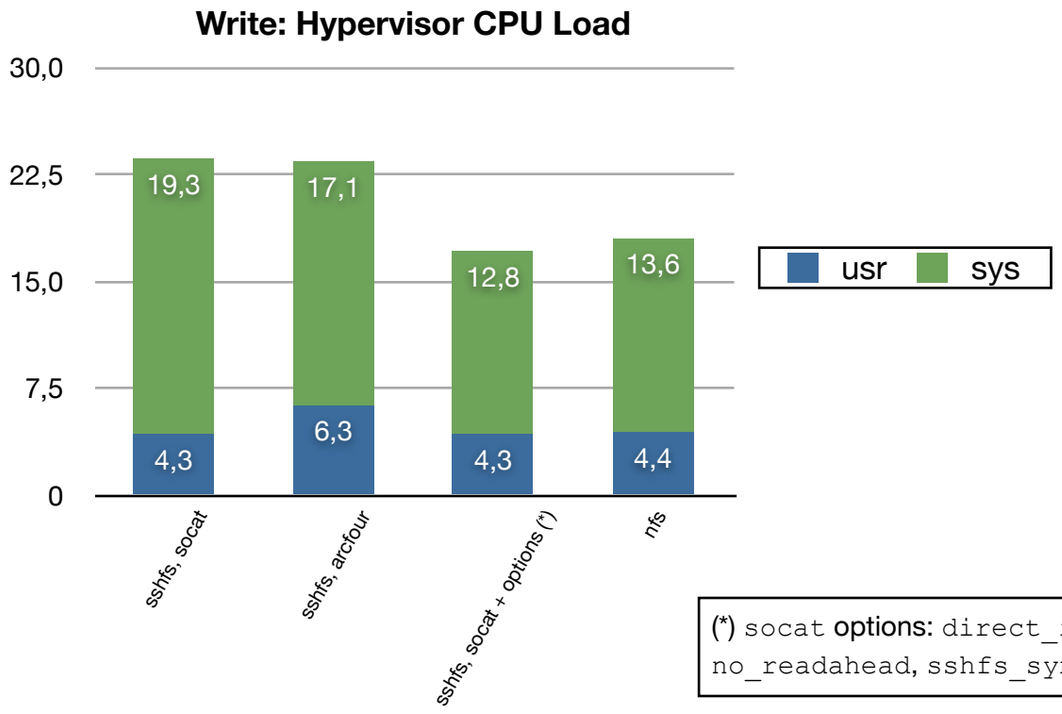


`sshfs` vs. `nfs`: throughput

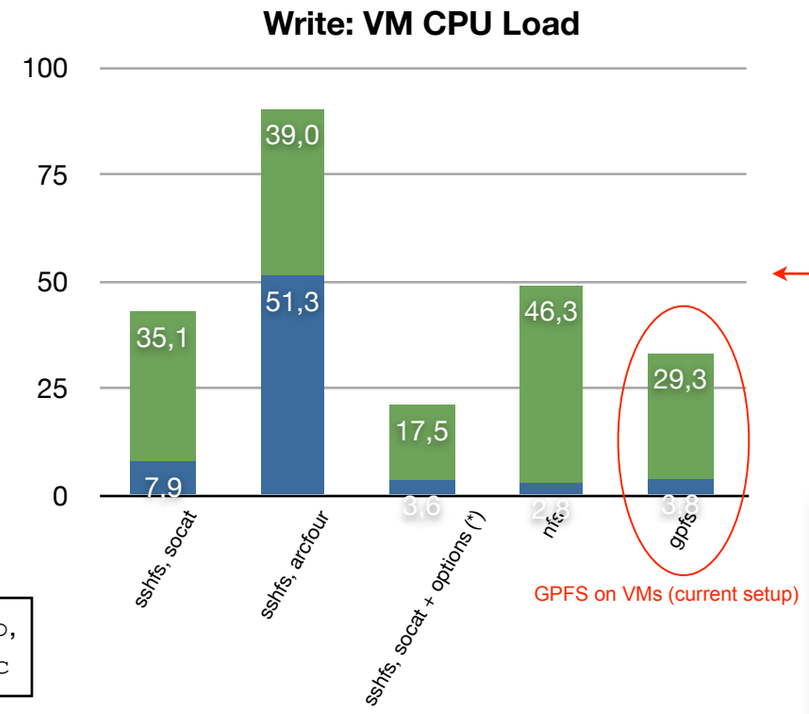
- `sshfs` throughput constrained by encryption (even with the lowest possible encryption level)
- Marked improvement (throughput better than `nfs`) using `sshfs` with *no encryption* through `socat`, esp. with some tuning
  - File permissions are not straightforward with `socat`, though - complications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



sshfs vs. nfs: CPU usage

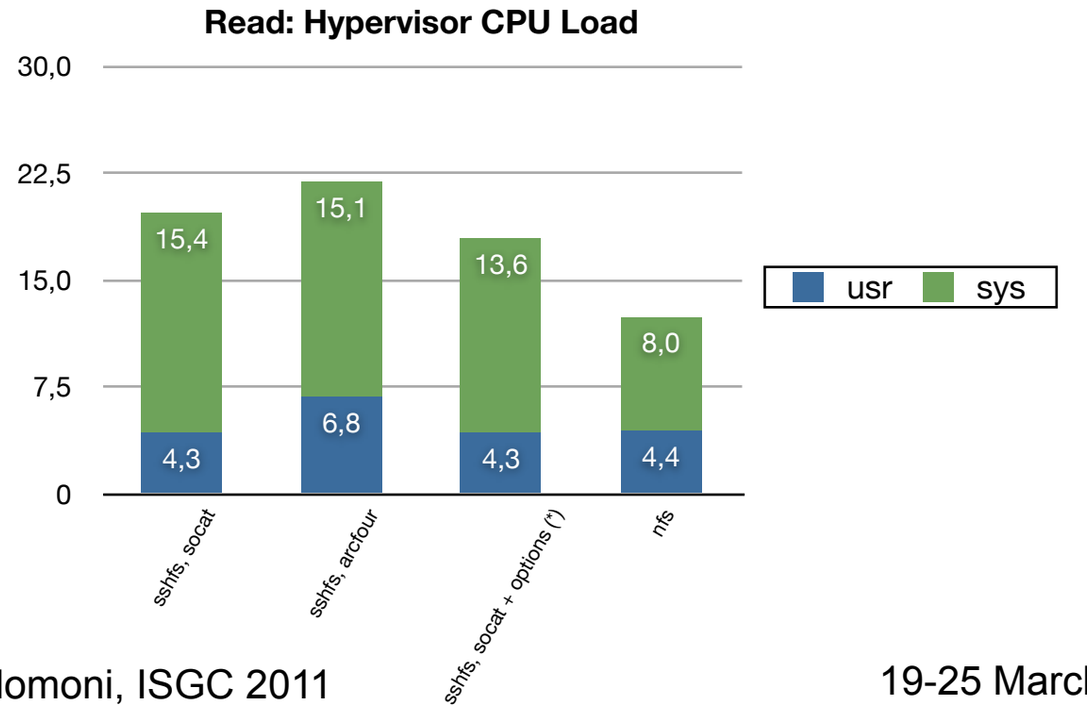


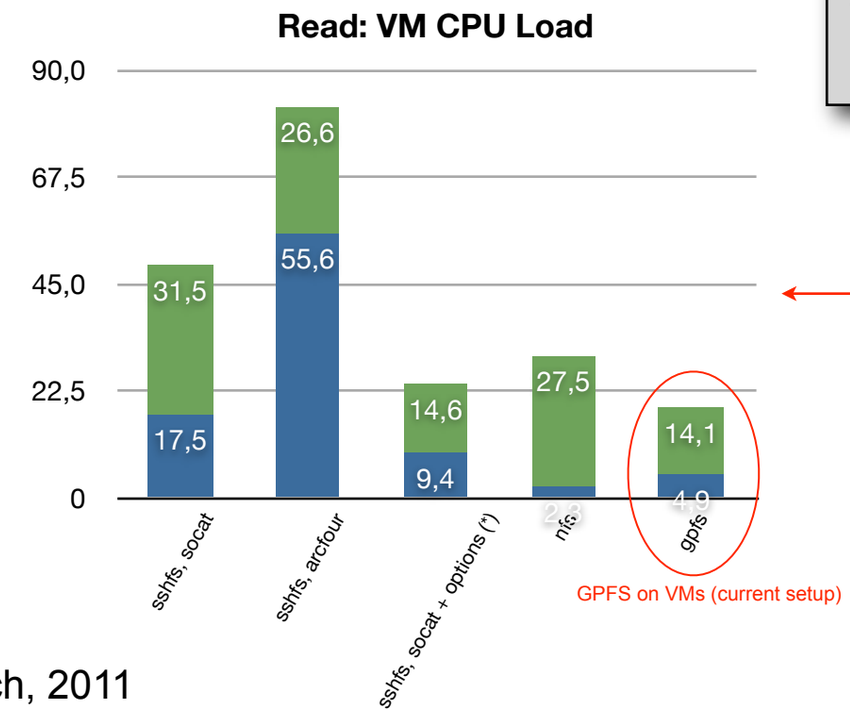
(*) `socat` options: `direct_io`, `no_readahead`, `sshfs_sync`



Write

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





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, especially in case of e.g. `glexec`-based identity change
- 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)
  - Avoid that network traffic between hypervisors and VMs exits the physical hardware using locally known address space and routing rules
- Support for `sshfs` or `nfs` gateways is scheduled to be included in WNoDeS 2 “Harvest”

Outline

- Introduction to WNoDeS
- Scaling locally distributed storage to thousands of VMs
- WNoDeS VM performance improvements
- Conclusions

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
- For all tests: since SL6 was not available at the time of testing, we used RHEL 6
  - Classic HEP-Spec06 for CPU performance
  - `iozone` to test 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, but on single Gigabit ethernet interfaces only, where its performance enhancements were not apparent. Tests on 10 Gbps cards are ongoing, and there we expect to see some improvements, especially in terms of latency.
  - 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

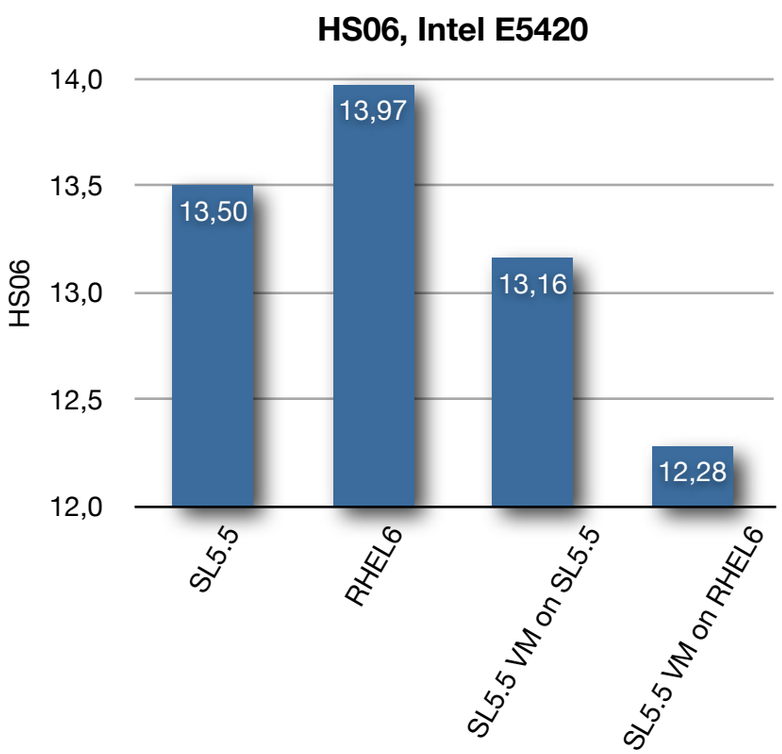
- HW: 4x Intel E5420, 16 GB RAM, 2x 10k rpm SAS disk using an LSI Logic RAID controller
- SL5.5: kernel 2.6.18-194.32.1.el5, kvm-83-164.el5_5.9
- RHEL 6: kernel 2.6.32-71, qemu-kvm 0.12.1.2-2.113
- SR-IOV: tests on a 2x Intel E5520, 24 GB RAM with an Intel 82576 SR-IOV card

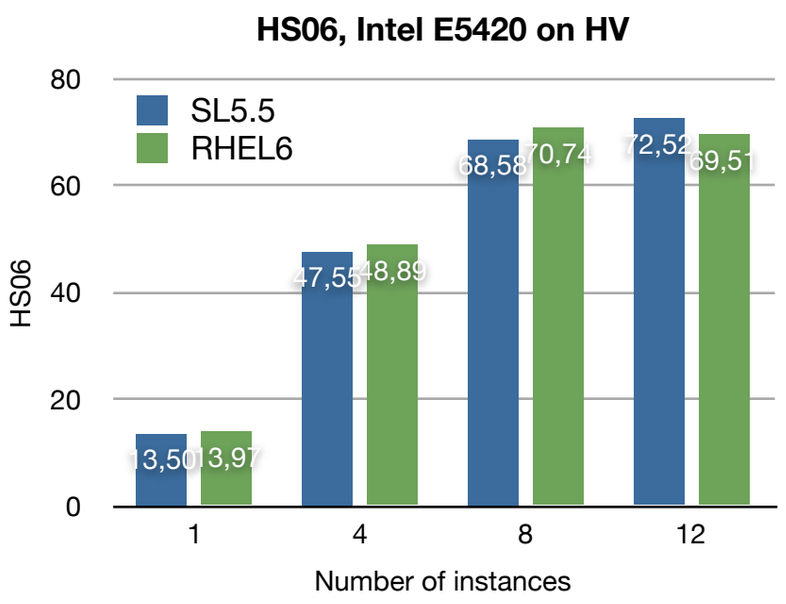
- iozone:

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

HS06 on Hypervisors and VMs (Intel E5420)

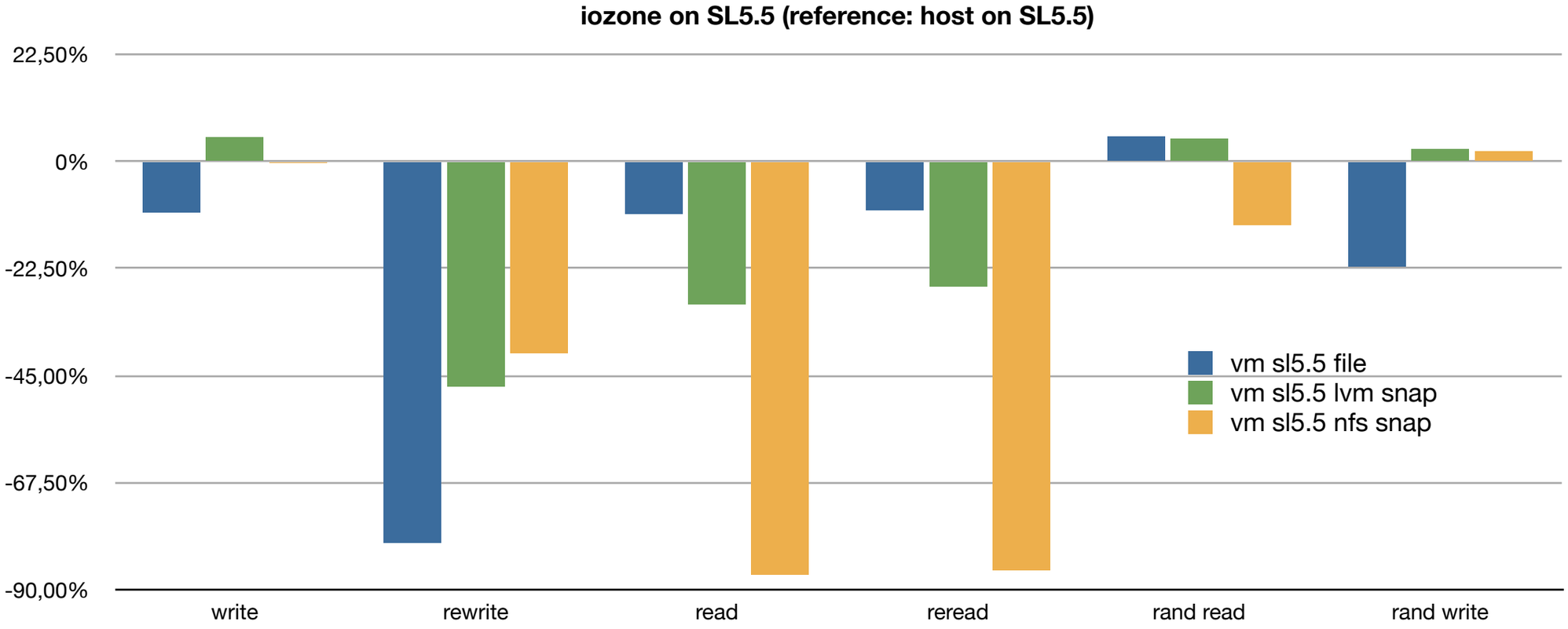
- Slight performance increase of RHEL6 vs. SL5.5 on the hypervisor
  - Around +3% (exception made for 12 instances: -4%)
- Performance penalty of SL5.5 VMs on SL5.5 HV: -2.5%
- Unexpected performance loss of SL5.5 VMs on RHEL6 vs. SL5.5 HV (-7%)
  - Test to be completed with multiple VMs





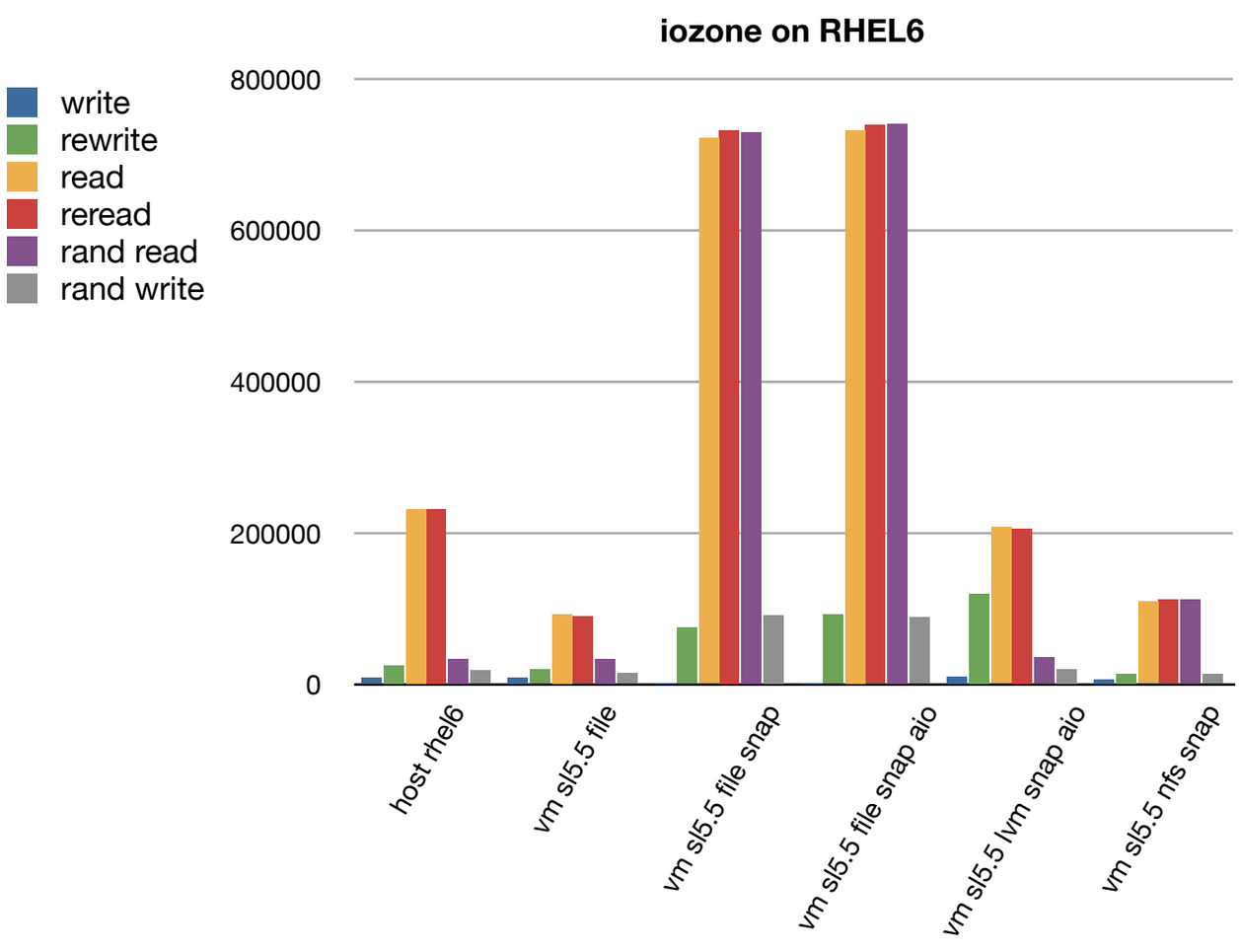
`iozone` on SL5.5 (SL5.5 VMs)

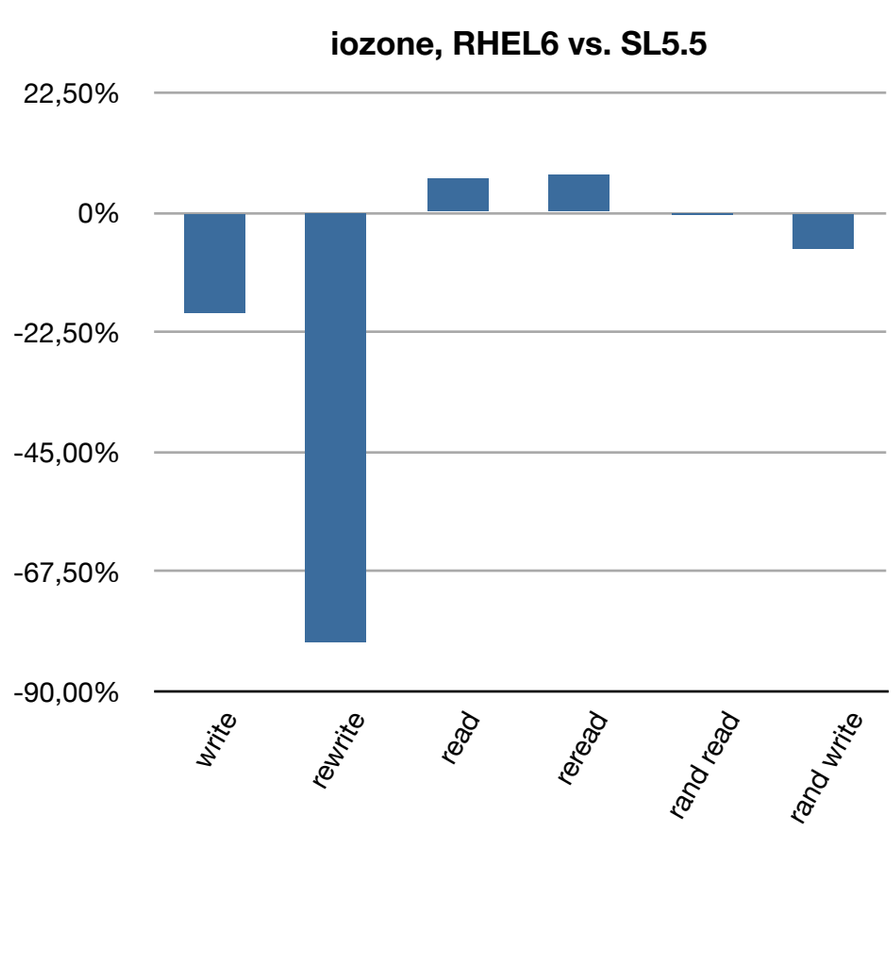
- `iozone` tests with caching disabled, file size 4 GB on VMs with 2GB RAM
- host with SL5.5 taken as reference
- VM on SL5.5 with just `-snapshot` crashed
- Based on these tests, WNoDeS will support `-snapshot` for the root partition and a (dynamically created) native LVM partition for `/tmp` and for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



`iozone` on RHEL6 (SL5.5 VMs)

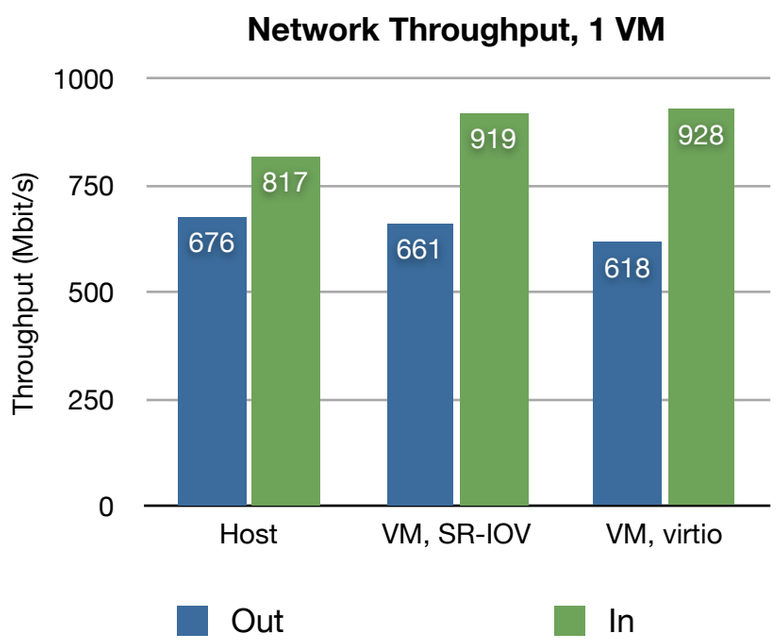
- Consistently with what was seen with some CPU performance tests, `iozone` on RHEL6 surprisingly performs often worse than on SL5.5
  - RHEL6 supports native AIO and `preadv/pwritev`: group together memory areas before reading or writing them. This is maybe the reason for some funny results (unbelievably good performance) of the `iozone` benchmark.
- 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 RHEL6/SL6 until we are able to get reasonable results in this area

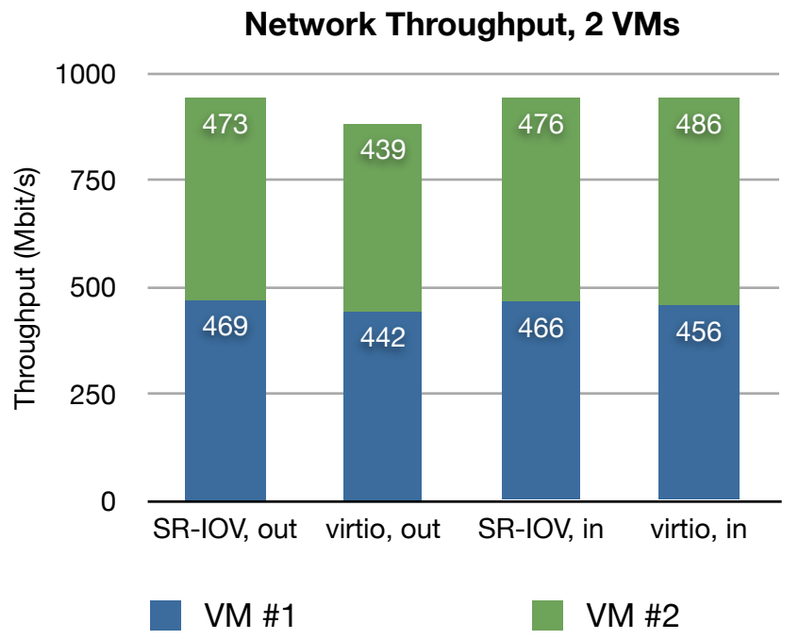


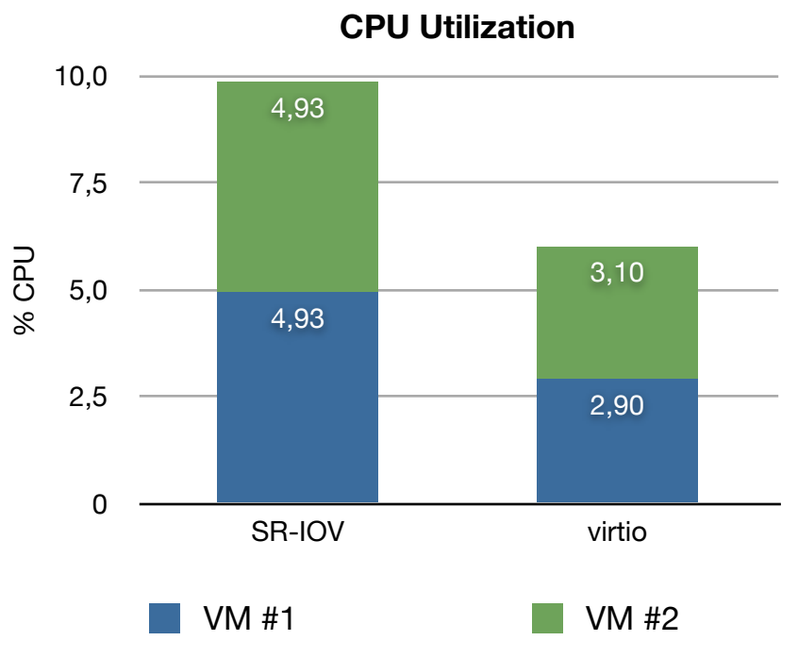


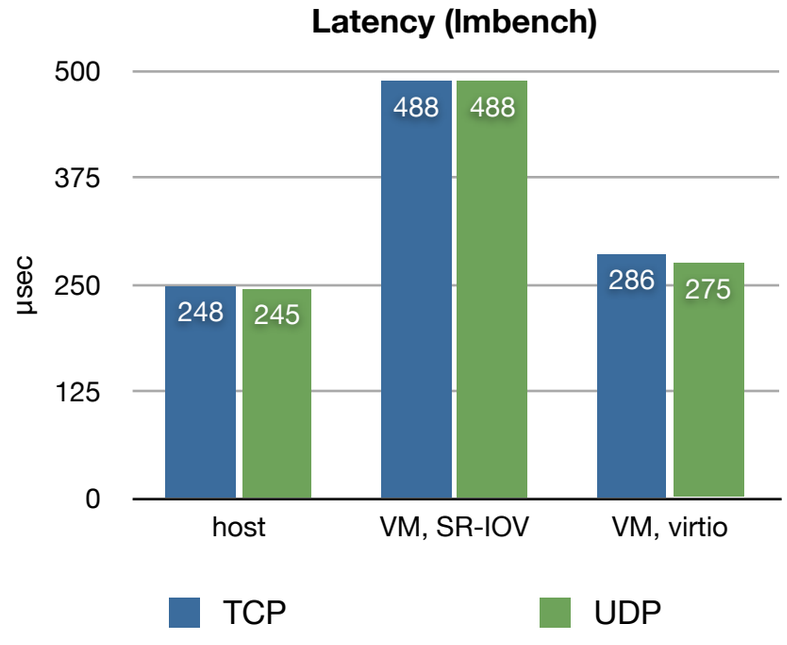
Network

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









WNoDeS VM Performance Improvements: Conclusions

- The `-snapshot` KVM flag is handy, but may incur in massive I/O overhead (or even crashes, with very large files)
- `-snapshot` is not directly supported by `libvirt`
  - Simple workaround integrated in WNoDeS
- Keeping the `-snapshot` flag and adding a dynamically-created LVM partition as a secondary VM disk maintains flexibility and significantly improves performance
  - Isolating I/O VM space
  - For security reasons, Cloud-based instances may need to use a completely separated partition
    - Needed also to support future “custom images”
- Direct support for dynamic LVM partitioning will be included in WNoDeS 2 “Harvest”
  - Flexible partitioning consistent with the WNoDeS definition of VM instance types (see talk on the WNoDeS Cloud Portal)

XML definition for `libvirt`-based WNoDeS VMs supporting the `-snapshot` flag:

```
  ...
  <devices>
    <emulator>/usr/local/bin/qemu-kvm-snapshot</emulator>
  ...
```

```
[davide@iz4ugl WNoDeS]$ cat /usr/local/bin/qemu-kvm-snapshot
#!/bin/bash

CMDLINE=
for i in $*
do
  if [[ $i =~ "^file=" ]]
  then
    if [[ $i =~ "boot=on" ]]
    then
      CMDLINE="$CMDLINE $i"
    else
      CMDLINE="$CMDLINE $i,snapshot=off"
    fi
  else
    CMDLINE="$CMDLINE $i"
  fi
done

exec /usr/libexec/qemu-kvm $CMDLINE -snapshot
[davide@iz4ugl WNoDeS]$
```

Outline

- Introduction to WNoDeS
- Scaling locally distributed storage to thousands of VMs
- WNoDeS VM performance improvements
- Conclusions

Conclusions

- VM performance tuning still requires detailed knowledge of system internals and sometimes of application behaviors
  - Testing is deliciously complicated
  - 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) to overcommit memory. Due to the nature of our typical applications, we normally do not overcommit memory (YMMV).
    - 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)
  - 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).
- 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.