Increasing performance in KVM virtualization within a Tier-1 environment

Andrea Chierici, Davide Salomoni
INFN-CNAF
andrea.chierici@cnaf.infn.it, davide.salomoni@cnaf.infn.it

Introduction

This work describes the optimizations we have been investigating and implementing at the KVM virtualization layer in the INFN Tier-1 at CNAF (Bologna, Italy), based on more than a year of experience in running thousands of virtual machines 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 (mostly related to High-Energy Physics). This work has been driven by the project called **Worker Nodes on Demand Service** (WNoDeS) [1], 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.

Testbed description

The testbed we used consists of two identical machines with dual Intel Xeon 5620 processor @2.4GHz, 24GB of DDR3 RAM @1066MHz, 2 Western Digital disks 300GB 10K RPM 16MB cache 2.5" SATA 3.0Gb/s Internal Enterprise Hard Drive and a 82574L 1Gbit/s LAN adapter by Intel. We added to this configuration a 160GB SATA II MLC Solid State Drive and a 82599eb 10Gbit/s LAN adapter, both by Intel. All disk I/O tests have been done using iozone, while network ones with netperf and lmbench.

DISK I/O

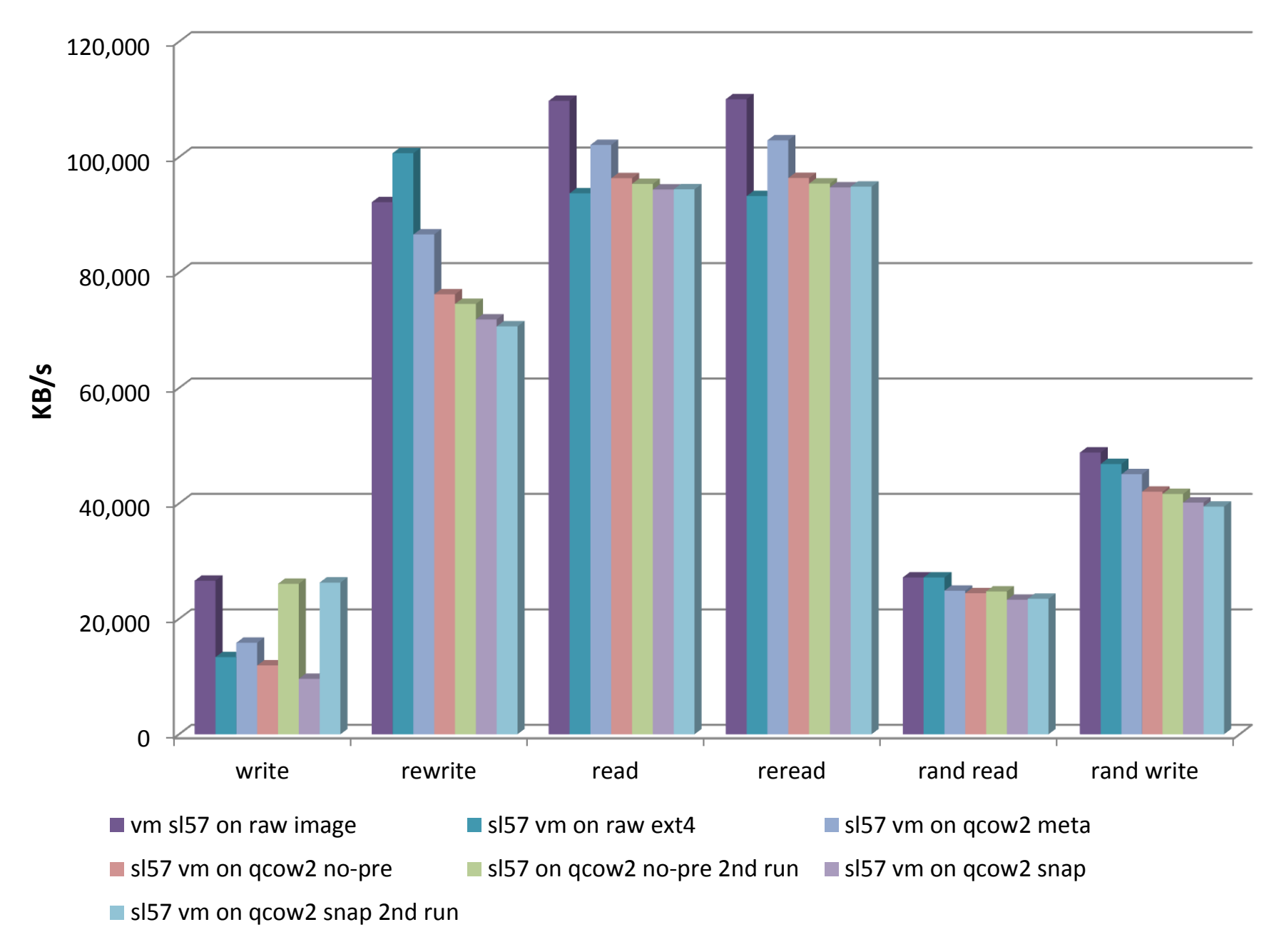
Disk I/O is a major issue in virtualization technologies. WNoDes currently uses KVM-based VMs, exploiting the KVM *-snapshot* flag. This allows to download (via either http or posix I/O) a single read-only VM image to each hypervisor, and run VMs writing automatically purged delta files only. This saves substantial disk space and time, which would be needed if one had to locally replicate multiple images. Performance characteristics of this solution (disk caching does not allow us to publish a reliable benchmark) and the latest enhancements in *qcow2* image handling pushed us to see if we can improve on this solution.

What we tested

WNoDeS is currently used to virtualize EMI [2] Worker Nodes, so the operating system of the VM we tested is uniquely SL5. In the future, when EMI WNs are available on SL6, we will update our test suite. We investigated performance differences among these solutions:

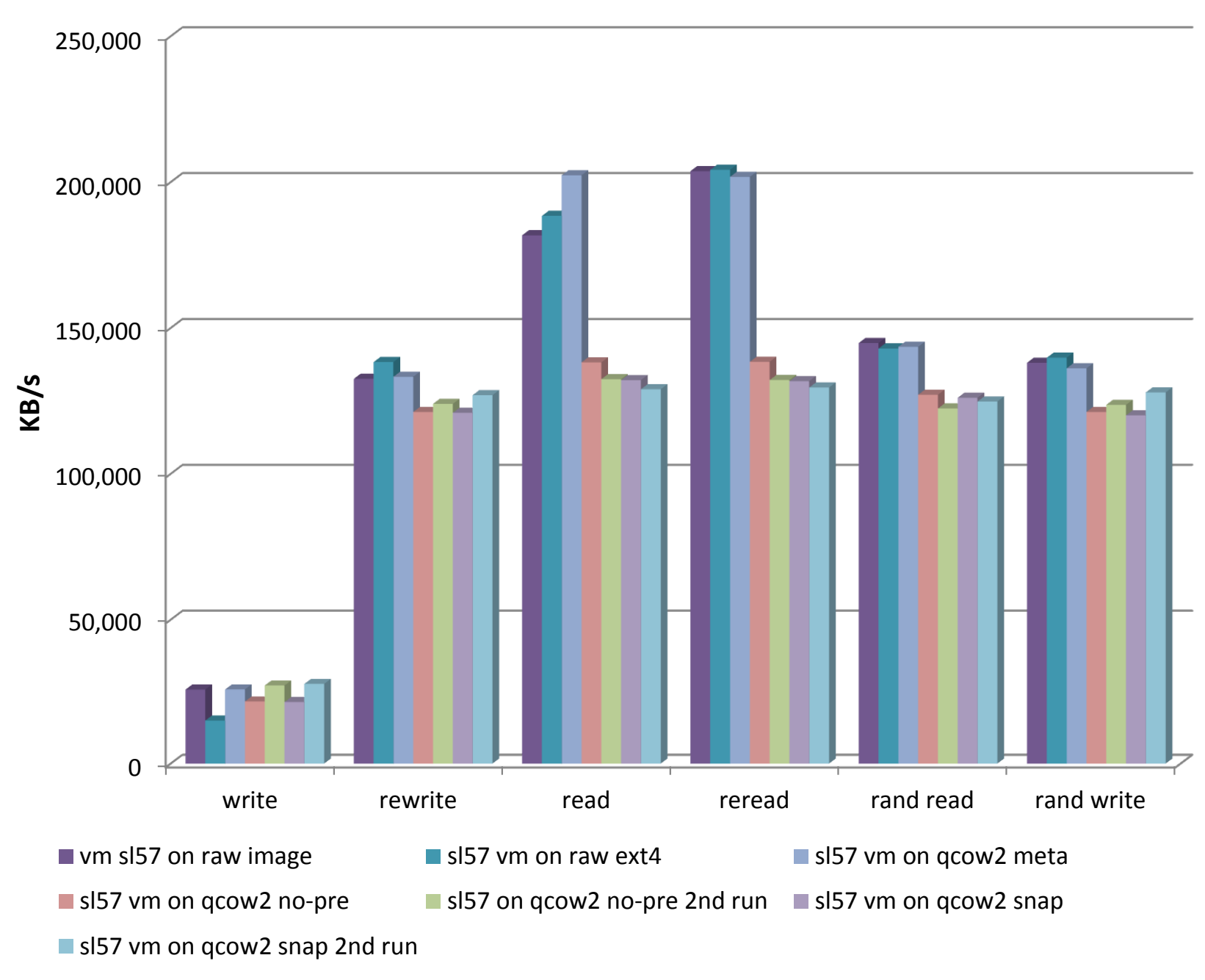
- **Ext4** file system: the default FS for SL6, declared to be faster and more reliable than the classic ext3. Since it's been back-ported to SL5, we wanted to test if using ext4 as FS for virtual image can improve performance;
- qcow2 with **metadata pre-allocation**: qcow2 image file with some internal image setup to speed-up I/O performance;
- qcow2 image **snapshot**: allows to create a new image that refers to an original image (so called a backing file) using Redirect-on-Write [3]; any changes to the snapshot will not be reflected in the original image.

Figure 1: Hard Disk Drive I/O performance



As can be seen by figure 1, if one excludes write performance, the remaining measures are rather similar. Writing performance is problematic and the difference between various solutions is more than 50% in some cases. The solution of qcow2 with metadata pre-allocation did not behave as expected for what concerns write speed and exhibited a significant performance degradation. A promising solution for the future of WNoDeS may be to use qcow2 snapshot with backing file. Indeed in the second run of the test, write speed was the same as raw image, providing a significant boost if compared to other solutions investigated. The main problem is the fact that we need to allow the disk snapshot to expand in order to reach good results.

Figure 2: Solid State Drive I/O perfomance



In figure 2, results similar to previous are illustrated running VMs on an SSD disk. With this kind of disk it's easy to see the general performance boost among all solutions. In this case qcow2 with metadata pre-allocation behaves as expected, in some cases even better than raw image. The most interesting solution even in this test is the one with backing file on qcow2 image. Indeed we put the snapshot on the ssd but the backing file was still on the hdd as in previous test. The performance boost is really impressive, and there is no need to allow the snapshot to expand to reach maximum write speed as in previous test.

From this bunch of tests we can claim that I/O virtualization has improved a lot with the latest kvm provided on SL6.1. Raw image is still appealing for general purpose but qcow2 format is more appealing for its enhanced features. Ext4 FS is clearly not mature enough on sl5 and so we discourage its adoption on production worker nodes. Current WNoDeS solution adopting *snapshot* option (that provides performance similar to raw image) could be substituted by qcow2 snapshot with backing file, particularly if adopting an ssd disk to store delta files (qcow2 snapshots).

SR-IOV

SR-IOV [4] devices can share a single physical port with multiple virtualized guests. Virtual Functions have near-native performance and provide better performance than para-virtualized drivers and emulated access. Virtual Functions provide data protection between virtualized guests on the same physical server as the data is managed and controlled by the *hardware* and not in software. These features allow for increased virtualized guest density on hosts within a data center. In other words, SR-IOV is able to better utilize the bandwidth of devices with multiple guests.

We performed several tests to verify if SR-IOV enabled network cards are really performing as vendors claim. Our test focused on a 10Gbps network card. We measured inbound and outbound connectivity, as well as latency.

Figure 3: Network Throughput

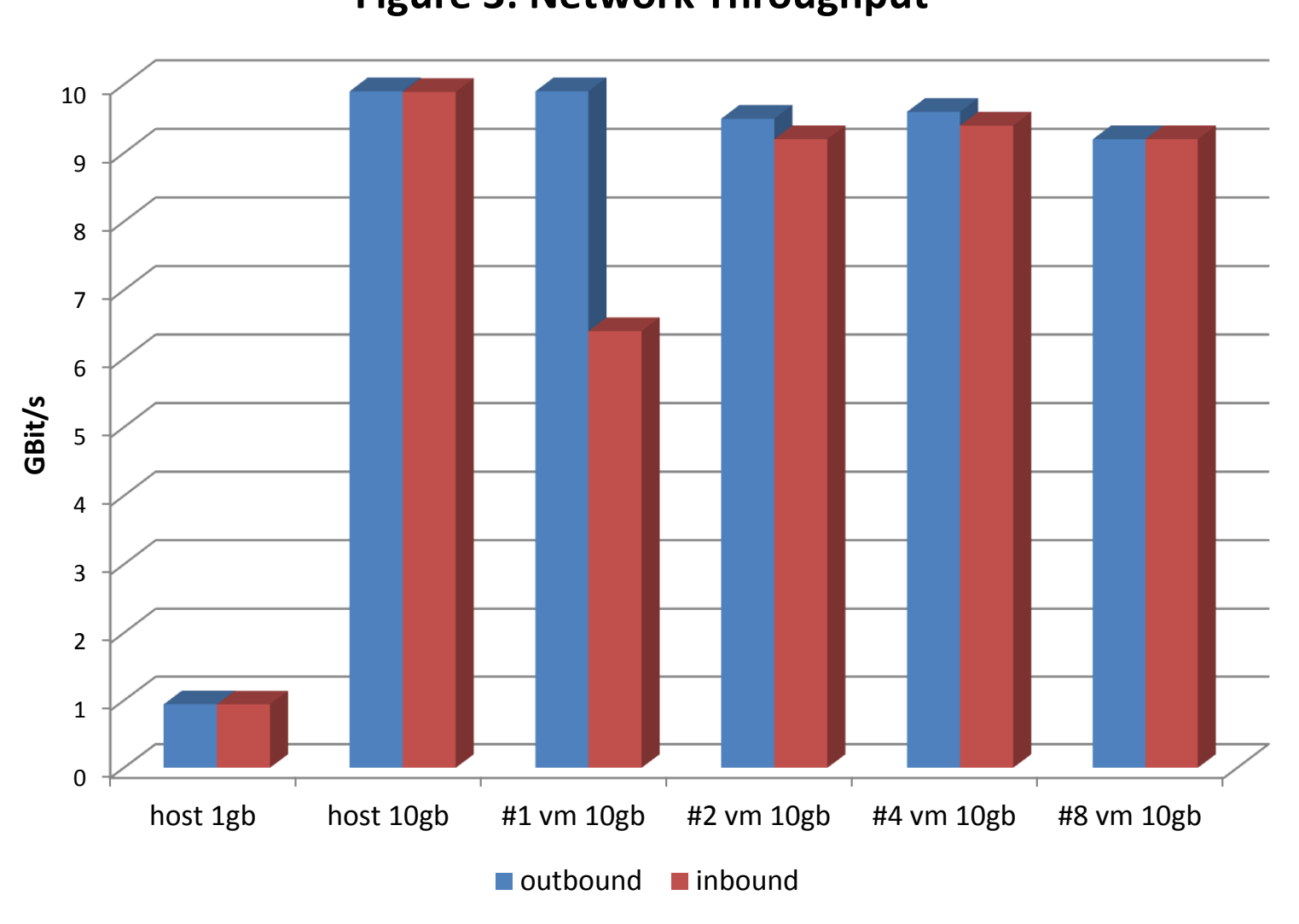
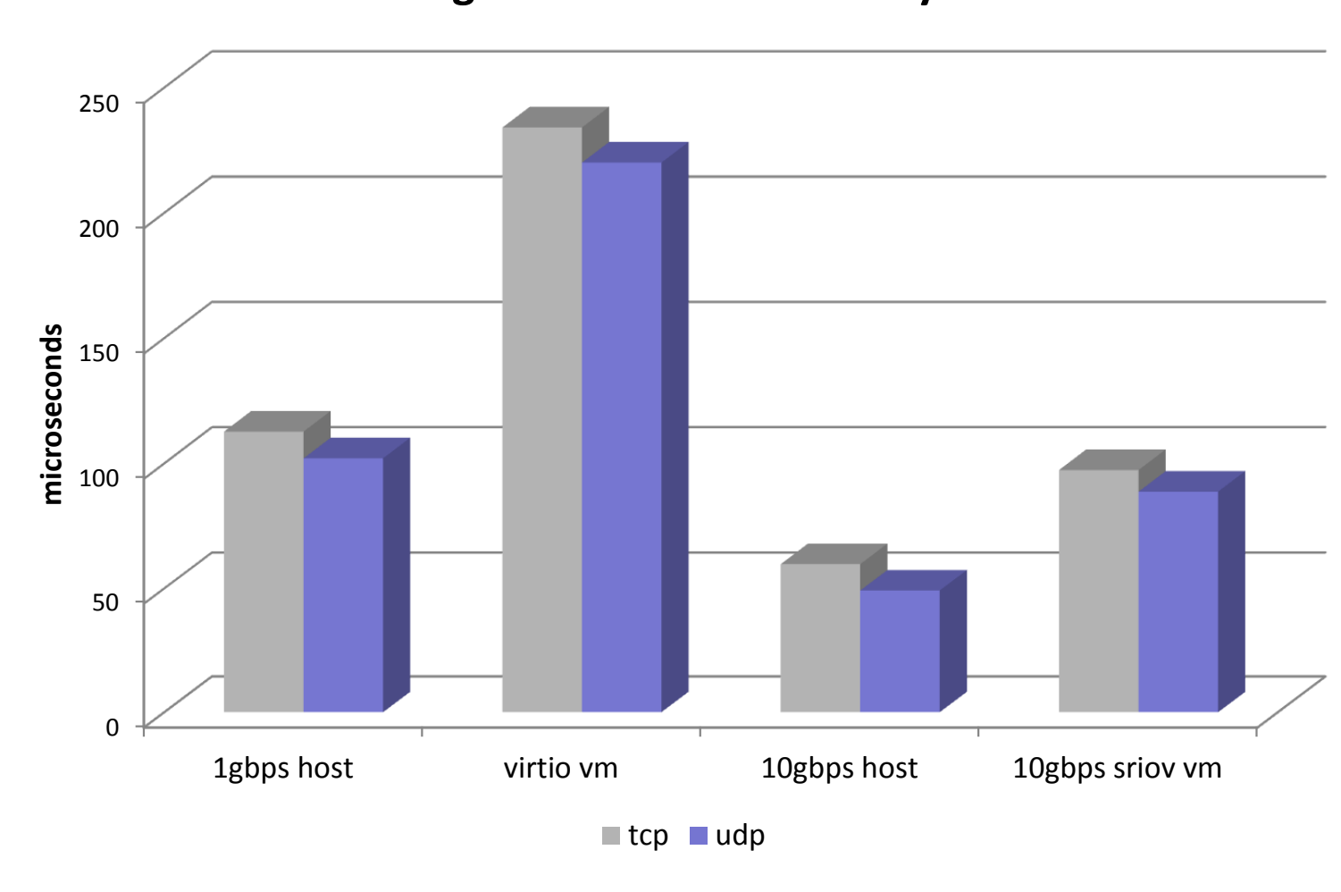


Figure 4: Network Latency

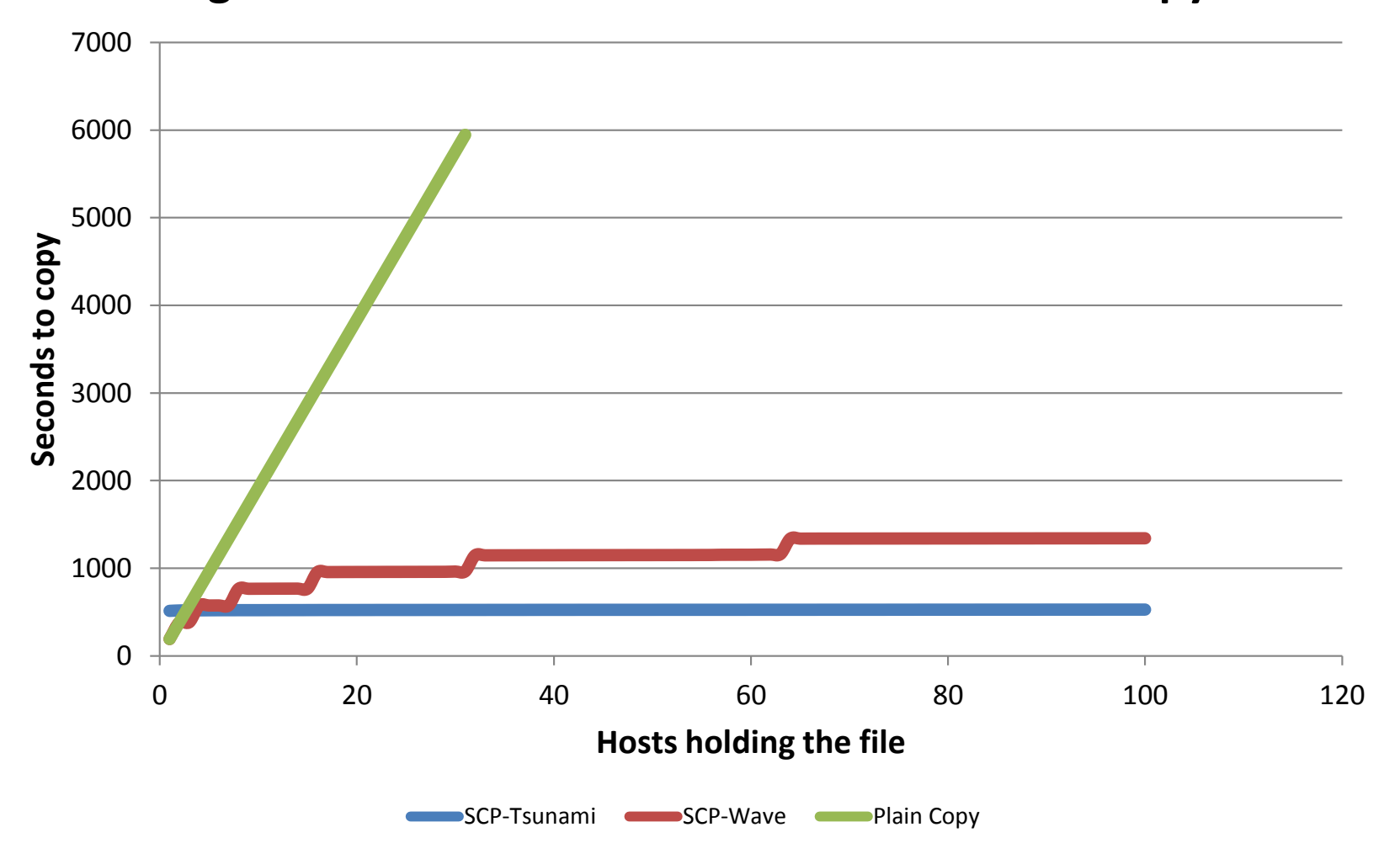


The 10Gbps SR-IOV NIC provides excellent performance, both for latency and aggregate throughput. As can be seen, there is no significant difference in throughput between VMs and host, except for unexpected low performance when running a single VM instance, to be investigated further. Network latency is close to bare metal hardware, 3 times better than virtio.

Image distribution

An important topic in the WNoDeS virtualization environment is the need to distribute the same image file to multiple virtualization hosts. Currently no specific tool is used, creating a potential bandwidth problem every time a new image has to be deployed on hypervisors. We tested several solutions and the most functional one is currently "scp-tsunami". **scp-tsunami** [5] is a Python script that splits the file/images in chunks and transfers multiple chunks between virtualization hosts.

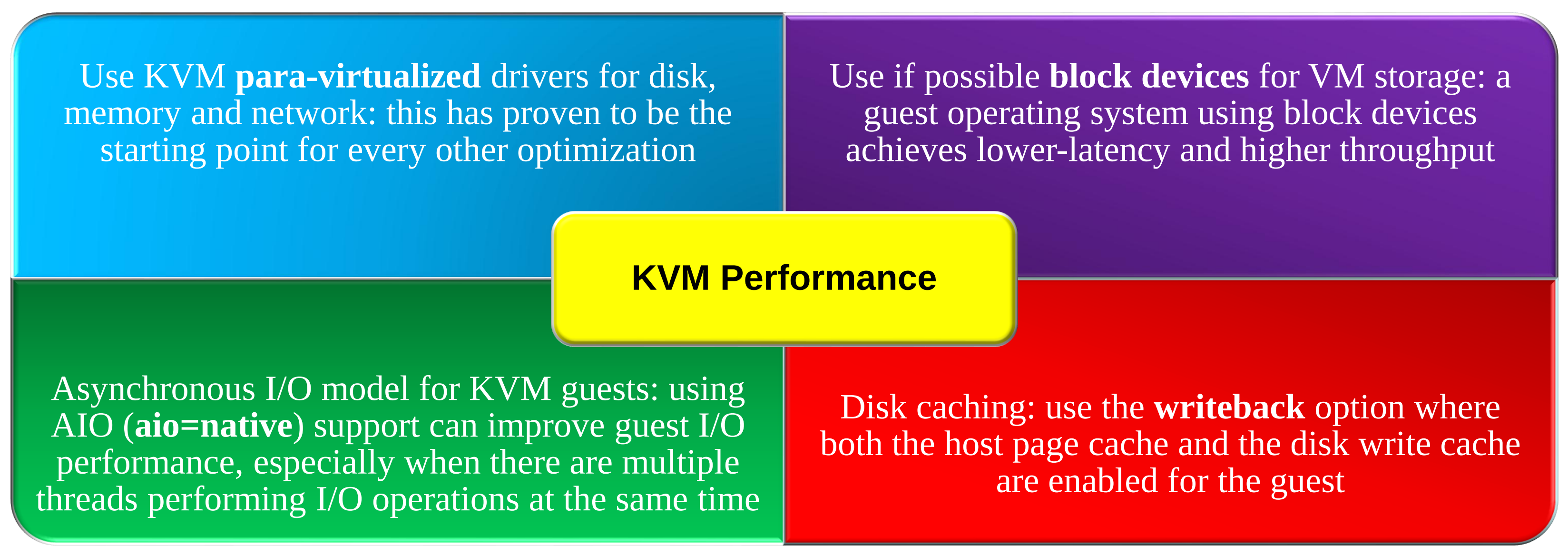
Figure 5: SCP-Tsunami VS SCP-Wave VS Plain Copy



With this simple script it's possible to pre-stage a VM image to a large set of nodes efficiently. Scp-tsunami resembles the bittorent protocol but does not require the same complicate setup. Scp-tsunami is a major improvement to scp-wave which offers a logarithmic speed-up, not enough for our production environment. In figure one can see the remarkable performance boost compared to the current solution adopted by WNoDeS (plain image copy). Every node owning the image file contributes to spread the image to others, drastically reducing the time required to complete the copy operation.

KVM best practices

Several documents are available on the Web suggesting best practices for the optimization of KVM (see for example [6]). Here we highlight them according to our personal experience in particular for the virtualization of EMI Worker Nodes.



Use KVM **para-virtualized** drivers for disk, memory and network: this has proven to be the starting point for every other optimization

Use if possible **block devices** for VM storage: a guest operating system using block devices achieves lower-latency and higher throughput

KVM Performance

Asynchronous I/O model for KVM guests: using AIO (**aio=native**) support can improve guest I/O performance, especially when there are multiple threads performing I/O operations at the same time

Disk caching: use the **writeback** option where both the host page cache and the disk write cache are enabled for the guest

References

1. http://web.infn.it/wnodes/index.php/wnodes
2. http://eu-emi.eu
3. http://www.ibm.com/developerworks/tivoli/library/t-snaptsm1/index.html
4. http://www.pcisig.com/specifications/iov/single_root/
5. https://code.google.com/p/scp-tsunami/
6. http://publib.boulder.ibm.com/infocenter/lnxinfo/v3r0m0/topic/liaat/liaatbestpractices_pdf.pdf