Design and implementation of a reliable and cost-effective cloud computing infrastructure: the INFN Napoli experience

V Capone$^{1,2}$, R Esposito$^2$, S Pardi$^2$, F Taurino$^{2,3}$, G Tortone$^2$

1 Università degli Studi di Napoli Federico II – Napoli, Italy
2 INFN-Napoli -Campus di M.S.Angelo Via Cinthia – 80126, Napoli, Italy
3 CNR-SPIN - Campus di M.S.Angelo Via Cinthia – 80126, Napoli, Italy

ecapone@na.infn.it, resposit@na.infn.it, spardi@na.infn.it, taurino@na.infn.it, tortone@na.infn.it

Abstract. Over the last few years we have seen an increasing number of services and applications needed to manage and maintain cloud computing facilities. This is particularly true for computing in high energy physics, which often requires complex configurations and distributed infrastructures. In this scenario a cost effective rationalization and consolidation strategy is the key to success in terms of scalability and reliability. In this work we describe an IaaS (Infrastructure as a Service) cloud computing system, with high availability and redundancy features, which is currently in production at INFN-Naples and ATLAS Tier-2 data centre. The main goal we intended to achieve was a simplified method to manage our computing resources and deliver reliable user services, reusing existing hardware without incurring heavy costs. A combined usage of virtualization and clustering technologies allowed us to consolidate our services on a small number of physical machines, reducing electric power costs. As a result of our efforts we developed a complete solution for data and computing centres that can be easily replicated using commodity hardware.

Our architecture consists of 2 main subsystems: a clustered storage solution, built on top of disk servers running GlusterFS file system, and a virtual machines execution environment. GlusterFS is a network file system able to perform parallel writes on multiple disk servers, providing this way live replication of data.

High availability is also achieved via a network configuration using redundant switches and multiple paths between hypervisor hosts and disk servers. We also developed a set of management scripts to easily perform basic system administration tasks such as automatic deployment of new virtual machines, adaptive scheduling of virtual machines on hypervisor hosts, live migration and automated restart in case of hypervisor failures.

1. Introduction
Over the last few years we have seen an increasing number of services and applications required to manage and maintain cloud computing facilities. This is particularly true for computing in high energy physics which often requires complex configurations and distributed infrastructures, and even more when the site is part of WLCG [1] or other computing grid in general. In addition, the continuously
diminishing of the funding is another factor that we have to face, and is foreseeable that this won't change at least in the next 3-5 years. In this scenario a cost effective rationalization and consolidation strategy is the key to success in terms of scalability, reliability and cost savings. With the growing density of current servers’ hardware, that allows to pack more than 48 computing cores (with the proper amount of system RAM) and 4/8 TByte of disk space in a single rack unit, the main path to follow is a massive use of virtualization to allow the complete exploitation of all this computing power.

The main goal we intended to achieve was a simplified method to manage our computing resources and deliver reliable user services, reusing existing hardware without incurring heavy costs. A combined usage of virtualization and clustering technologies allowed us to consolidate the most part of the data centre services on a small number of physical machines, reducing electric power costs. As a result of our efforts we developed a complete solution for data and computing centers that can be easily replicated using commodity hardware.

In this work, we describe an IaaS (Infrastructure as a Service) cloud computing system, with high availability and redundancy features, which is currently in production at INFN-Napoli and ATLAS Tier-2 data centre.

Fig. 1 - The hardware and network topology.

2. Initial system setup
The design we had in mind was composed by a pool of servers used as hypervisors, with the sufficient computing power (CPU cores and RAM), and of a storage area where to put all the disk images of the VMs; for the physical separation between the hypervisors and the VM files, the network connectivity
should have been properly designed, to provide good performance but also resiliency and fault
tolerance.

2.1. Hardware components
We started from commodity hardware we already owned, that was upgraded in order to fulfill the
requested performance. In particular, three Dell PowerEdge 1950 rack servers have been used as VMs
hypervisors, and two Dell PowerEdge 2950 as VM stores.

All servers are equipped with dual Intel Xeon E5430, providing 8 cpu cores per server, with 8
Gbyte of RAM. The upgrades consisted in 8 Gbyte RAM and a 2 ports Ethernet NIC on every
hypervisors, 6 x 1.5 Tbyte SATA hard disks and a 4 ports Ethernet NIC on both storage servers. The
storage server disks were configured in RAID5 (dm-raid software mode); we chose to not use a
RAID6 setup because the complete mirroring of the servers (added to the RAID5) was considered to
be at a sufficient level of fault tolerance, so we could use a storage space of 7.5 TByte instead of 6
Tbyte.

Hypervisor servers had 2 x 500 Gbyte disks configured in RAID1, that was adequate for our needs,
given the fact that this space was used solely for OS installation.

Furthermore a dedicated 24 gigabit ports Cisco Catalyst 2960G switch was added to the hardware
configuration: the reason and usage mode for this will be explained in the following sections.

2.2. Software and Operative Systems
The OS used on all servers was initially Scientific Linux 5.5, with KVM [2] as virtualization system.
Before selecting Linux KVM, we evaluated VMWare ESXi and Xen, two other virtualization
platforms providing similar capabilities, but the first, although being free, is not an Open Source
project, and the latter is not fully integrated in the Linux kernel. We selected KVM as the best
architecture for virtualization on modern processors with fast hardware virtualization support (VT-x
and NPT on Intel or AMD-V and EPT on AMD). This hypervisor is built into mainline Linux kernel,
fully integrated in most distributions and is the foundation of commercial products like RedHat
Enterprise Virtualization [3] and open-source project like OpenStack [4] or Eucalyptus [5].

After a few months, we updated all servers to Scientific Linux 6.2 to use the new KVM version and
KSM (Kernel Samepage Merging) [6], a memory deduplication feature that enables more guests to
share the same memory pages of the host. The chosen solution for the storage subsystem was
GlusterFS; more on this in the following sections.

Virtual machines have their root file systems on qcow2 disk-image files, stored on the GlusterFS
network file system.

The OSs of the guests running on the infrastructure are mostly Linux-based (especially Scientific
Linux), but also some FreeBSD and Windows Server installation is present.

2.3. Network topology
The main requirements for the network serving our infrastructure are: performance, reliability and
resiliency. To achieve these goals, we set up a double path between every hypervisor and both storage
servers, with two different switches involved, so that the failure of one of them doesn’t impact on the
execution of the Virtual Machines, whose disk images are hosted on the remote storage servers.

The Cisco Catalyst 6509 is the core switch of our science department network infrastructure, and
also the concentration switch of the central services; our service data centre is not based on a standard
Top-of-rack structure, instead there are three dedicated 48 ports line card for the three racks in the data
centre, where all servers in the rack are directly connected with the correspondent switch line card
through a patch-panel. The central rack, that hosts the Cisco Catalyst 6509, is equipped with 3 patch-
panels.

Every hypervisor server is connected to it via the onboard dual gigabit Ethernet port, in LACP
bonding mode, so to provide the necessary connectivity and the sufficient bandwidth to the VMs: this
link is in trunk mode, so that every VM can be connected to the desired VLAN. A number of sub-
interfaces are created on every hypervisor host, one for every existing VLAN, that are mapped on the relative virtual bridge. The KVM hypervisor uses these virtual bridges to connect the NICs of the VMs. The use of the LACP mode, instead of the more effective Adaptive Load Balancing mode, is due to the fact that it's the only channel bonding mode that is compatible with virtual bridges mapped on multiple VLANs. Also the storage servers have this kind of network configuration, so that they can operate as hypervisor servers in case of emergency.

We needed to separate the network traffic related to hypervisor to/from storage data transfer from all other network traffic, because of the fact that from the performance and stability of these transfers depends the performance and stability of every VMs. For this reason a private VLAN hosts the data traffic between the storage servers and the hypervisors, and a second switch (Cisco 2960G) is connected to Cisco 6509 via a 3 x 1 Gbit LACP bond link; within this VLAN every storage server is connected with three gigabit links to the Cisco 2960G and the fourth to the Cisco 6509, while every hypervisor is connected with one link to both switches; the multiple connection of the servers to the two switches is achieved with the Adaptive Load Balancing mode, also because it's the only mode that permits to span the links between different switches. Within this topology, the Cisco 2960G is completely dedicated to the network traffic of the storage VLAN, while the Cisco 6509, other than as access switch towards the department LAN, as the redundant switch for the storage VLAN.

The use of the Cisco 2960G as the primary switch for the storage-related network traffic has a reason in performance too, because the Cisco WS-X6548-GE-TX 48 ports line-in cards that are installed on the Cisco 6509 have an oversubscription of 4:1, while the 2960G guarantees full wire-speed for all 24 ports.

3. The storage subsystem
We needed a reliable and open-source solution to design our storage as a fault tolerant and reasonably fast system; so we chose GlusterFS [7] as the best fit to our constraint and expectations, also given its relatively easy deployment and configuration.

GlusterFS is an open-source, clustered file-system for scaling the storage capacity of many servers to several petabytes. It aggregates various storage servers or bricks over Infiniband RDMA and/or TCP/IP interconnect into one large parallel network file system. This open-source filesystem, now owned by RedHat [8], is posix-compliant, enables a global namespace on clustered storage, is modular and stackable and has built in replication and self-healing features.

GlusterFS in its default configuration does not stripe the data, but instead distributes the namespace across all the servers. In terms of design, a small portion of GlusterFS is in the kernel and the remaining portion is in user space. The calls are translated from the kernel VFS to the user space daemon through the Filesystem in UserSpace (FUSE). Users can access application data and files in a global namespace using a variety of standard protocols.

Key features of GlusterFS:
• Modular, stackable storage OS architecture
• Data stored in native formats
• No metadata – Elastic hashing
• Automatic file replication with self-healing
A GlusterFS “brick” (a server and a directory path, like `fssrv1:/vmstorebrick`) can be combined, thanks to “translators” (modules chained together to move data), in “volumes” (collection of bricks with the same requirements). We setup two disk servers, each one with a brick on an ext3 filesystem, and the replica translator, to obtain a “network mirrored” volume used as storage backend for the guest disk images. The hypervisor hosts mount this volume on boot and send each read and write request to both servers, speeding up reading operations and keeping write operations secure.

4. Custom scripts

Integrated in Scientific Linux are some useful VM management tools, like `virt-manager`, a simple and easy to use graphical user interface, and the `virsh` command, a comprehensive program to create, pause, shutdown, migrate and list virtual machines, both based on the libvirt project [9]. On top of this command and libraries, we developed a set of CLI management scripts to easily perform basic system administration tasks such as automatic deployment of new virtual machines, adaptive scheduling of virtual machines on hypervisor hosts, live migration and automated restart in case of hypervisor failures. For example, to provision a new Linux guest, we use a command line like this:

```
# ./new-linux-vm.sh sl 56 20G 1024 newname na.infn.it x86_64 172.16.0.101 255.255.0.0 172.16.0.254
```

that asks our cloud to deploy a 64 bit Scientific Linux (sl) guest, version 5.6, with a 20 GB virtual disk, set the hostname to “newname”, the domain name to “na.infn.it” and the IP address to 172.10.0.101/16. This task is accomplished by modifying a kick-start template and uploading it on a web server, creating a new virtual machine definition file in xml and firing up the new guest (that will install and reboot in 3/4 minutes) on the lesser loaded physical server.

We can also list running guests:
```
[root@exec05 ~]# vm-list
Running on exec01:
bastion1 listman sl6test tino winamna
```

Running on exec02:
ina-srv1 leonardo mx1-fisica natter1 papercut

Running on exec03:
auth01 auth02 fan listserv proxy2 webdip

Running on exec04:
cassini dsna1 lxprint2 natterfis spin

Running on exec05:
dipsf dsna6 imap-fisica luxna2 tauwin01 tauwin02

Find which host runs a specific guest:
[root@exec05 ~]# vm-find leonardo
Domain leonardo is running on exec02

Live migrate a guest to a different host:
[root@exec05 ~]# vm-migrate sl6test exec04
Migrating domain sl6test from exec01 to exec04.............Done.

The system also periodically check the list of running guests, and also, thanks to our Nagios-based monitoring system [10], we always keep the cloud in a coherent and trouble-free state.

5. Performance and features
We’ve performed some IOzone [11] benchmark to see what I/O disk performance we could’ve expected, especially from r/w operations on the host disks.

![Disk performance (IOZone benchmark)](image)

Fig. 3 – Disk r/w performance to GlusterFS.

The first result is the throughput of one of the storage servers vs. the GlusterFS brick on its own disk array; the second is the throughput from the hypervisor host OS to the GlusterFS volume, and third is the throughput from the guest to its own disk image. While the first result relies solely on the RAID5 storage subsystem, with no network involved (just to put down the highest result achievable),
the second and third are also impacted by network performance. We’ve not reported CPU load in the various tests because they’ve been performed on different servers, so the results wouldn’t be comparable; and since the storage servers, on which GlusterFS mostly impacts, don’t have other significant concurrent tasks, the system load for them is not a main issue.

With regard to fault tolerance and high availability, our system has proved to be a very reliable solution. We’ve tried a variety of failing scenarios, causing different network, power and disk failures.

The VMs continue to work flawlessly in the following cases:

- Failing of all but one network link (port or cable) from hypervisor to the storage;
- Failing of all network links (port or cable) from hypervisor to the wan;
- Complete outage of one of the switches;
- Failing of one disk on the storage servers;
- Complete outage of one of the storage servers.

When the failing disk server returns to normal activity, GlusterFS self-healing integrated mechanism performs a background transparent reconstruction of missing replicas.

In other cases, the execution of the guests can be interrupted for a while, but the VM is back in execution in the time it takes for a reboot on a different hypervisor:

- Failing of both network links from hypervisor to the storage;
- Outage of the hypervisor where the guest was running on.

The outage of one of the hypervisors causes one of our custom scripts to migrate all the VMs running on it to another available host, starting from the one with the lesser load. The worst-case scenario that guarantees the continuity of all the services is:

- Outage of all hypervisors but one;
- Outage of one of the switches;
- Outage of one of the storage servers;
- Failing of one disk on the survived storage server;
- Failing of one disk on the survived hypervisor.

Even after this very unlikely chain of events, all the VMs will continue to be up (even if with reduced performance). Makes sense to point out that all our servers have a redundant power supply, that is connected to separated power lanes of two different UPS systems, and so is the Cisco Catalyst 6509. So also the power outage caused by the failure of one UPS is not a problem for the whole system.

### 6. Future improvements

We’re now acquiring a new set of servers to deploy a second and more powerful version of our private cloud system. Next step will be to virtualize all the remaining services that are still running on physical machines, like SMTP and IMAP mail servers, public login and users home directory systems. We’ve already improved the computing power of the current cloud with two more hypervisor servers, that have quite the same hardware specs of the former three.

The new hardware for the next version of our cloud will have a lot more available computing cores and storage space, as it will consist of four hypervisor servers, each with dual 12-core AMD CPUs and 48 Gbyte of RAM, and two storage servers with 12 x 3 Tbyte SATA hard disks each.

After the deployment of the new hardware, we will test some more advanced software system for the cloud management, like OpenStack and OpenNebula [12], to see if they fit our needs.

One last aspect we have to investigate is how to perform the snapshots of the VMs in a reliable manner, for the naïve method that comes with the qcow2 format causes a huge network and disk load on the storage servers, that can cause the timeout of some of the hypervisors in accessing the disk images, and then the sudden death of some of the VMs.
7. Conclusions
We have set-up a complete, cheap, reliable, easy to manage private cloud, with commodity hardware and open-source only software. The measured performances have shown that this kind of system, in the configuration we’ve realized, could well serve for applications that are not strongly I/O bound, because while it’s not a problem to provide more computing power or RAM to the hypervisor servers, the overall performances have a bottleneck in the disk I/O of the guests (due to network speed limits), that reaches about 55-60 MByte/s.
For this reason this system is more adequate to deploy services like DNS, DHCP, BDII, monitoring, or some kind of computation node (depending on the software needed), not to mention all other service nodes that are part of a grid site, like Computing Elements, DPM or STORM services, User Interfaces, and in general every grid service that is not directly storage-related (like the actual Storage Elements), but that is critical for the grid to properly work. For general purpose data center, almost every kind of central service can be deployed on our cloud solutions: SMTP and IMAP servers, authentication and RADIUS services, VPN, NAT, firewalls, and so on.
A different kind of Gluster configuration (based on multiple servers with striping+mirroring) and/or the use of a more powerful storage subsystem (not to mention the use of high-speed network ports between hypervisors and storage), should provide the boost in disk I/O performance that can make this kind of cloud system also fit for a grid storage subsystem, or in general for a Storage Area Network system. As soon as we’ll have the proper hardware available, tests will be made in this direction.

References
[1] WLCG: http://lcg.web.cern.ch/lcg/
[2] KVM: http://www.linux-kvm.org/page/Main_Page
[3] Red Hat Enterprise Virtualization: http://www.redhat.com/products/virtualization/
[4] OpenStack: http://www.openstack.org/
[5] Eucalyptus: http://www.eucalyptus.com/
[6] Kernel Samepage Merging: http://www.linux-kvm.org/page/KSM
[7] GlusterFS: http://www.gluster.org
[8] Red Hat to Acquire Gluster: http://www.redhat.com/promo/storage/brian-stevens.html
[9] Libvirt Virtualization API: http://libvirt.org/
[10] Nagios: http://www.nagios.org/
[11] IOzone Filesystem Benchmark: http://www.iozone.org/
[12] OpenNebula: http://opennebula.org/