Managing a tier-2 computer centre with a private cloud infrastructure

Stefano Bagnasco[1], Dario Berzano[1,2,3], Riccardo Brunetti[1,4] , Stefano Lusso[1] and Sara Vallero[1,2]

[1]Istituto Nazionale di Fisica Nucleare, Via Pietro Giuria 1, 10125 Torino, Italy
[2]Dipartimento di Fisica, Università degli Studi di Torino, Via Pietro Giuria 1, 10125 Torino, Italy
[3]CERN - European Organization for Nuclear Research, CH-1211 Geneva 23, Switzerland
[4]Now at Colt Engine S.r.l., Pianezza, Italy

E-mail: stefano.bagnasco@to.infn.it

Abstract. In a typical scientific computing centre, several applications coexist and share a single physical infrastructure. An underlying Private Cloud infrastructure eases the management and maintenance of such heterogeneous applications (such as multipurpose or application-specific batch farms, Grid sites, interactive data analysis facilities and others), allowing dynamic allocation resources to any application. Furthermore, the maintenance of large deployments of complex and rapidly evolving middleware and application software is eased by the use of virtual images and contextualization techniques. Such infrastructures are being deployed in some large centres (see e.g. the CERN Agile Infrastructure project), but with several open-source tools reaching maturity this is becoming viable also for smaller sites. In this contribution we describe the Private Cloud infrastructure at the INFN-Torino Computer Centre, that hosts a full-fledged WLCG Tier-2 centre, an Interactive Analysis Facility for the ALICE experiment at the CERN LHC and several smaller scientific computing applications. The private cloud building blocks include the OpenNebula software stack, the GlusterFS filesystem and the OpenWRT Linux distribution (used for network virtualization); a future integration into a federated higher-level infrastructure is made possible by exposing commonly used APIs like EC2 and OCCI.

1. Introduction

The Computer Centre at INFN Torino hosts a medium-sized Scientific Computing infrastructure that caters to the needs of a number of applications, the largest ones being a WLCG Tier-2 Computing Centre [1] for the ALICE experiment at the CERN LHC [2] and a PROOF-based Parallel Interactive Analysis Facility [3] for the same experiment. The centre is planned to grow in the near future by integrating more use cases, including among others a Tier-2 computing centre for the BES-III Experiment at IHEP (Beijing) and a small computational farm for the local Theoretical Physics group.

However, while the amount of resources and the variety of applications is steadily increasing, manpower unfortunately is not. It is thus becoming almost mandatory to consolidate such resources to achieve scalability and economies-of-scale not only by centralizing procurement but also and mainly by separating application management, that can be delegated to experienced users, from infrastructure management that can not. In this sense, Computing Centres need to become providers of computing and storage resources, not (only) of high level services.

Beyond being a trend and a buzzword in the IT world, the Cloud Computing paradigm actually promises to be a way to raise the level of abstraction and implement such separation. By converting our computing farm into a Cloud infrastructure according to the Infrastructure-as-a-Service (IaaS) paradigm, we were successful in reducing the amount of manpower needed by management tasks. Processes such as the dynamical reallocation of resources across the applications, the maintenance of large deployments of complex and rapidly evolving middleware and application software, are eased by the use of virtual images and contextualization techniques. At the same time the flexibility of the infrastructure was increased, allowing us to easily provide on-demand computing resources to a number of smaller applications and short-term requirements.

Furthermore, a number of larger-scale Cloud Computing project are starting in Italy and Europe to deploy large-scale distributed infrastructures: a local Cloud, provided its implementation is designed for interoperability, follows some existing standard, and exposes widely-used interfaces, should allow us to immediately take part in such activities.

Given these considerations, and taking into account the need not only to maintain but also to develop the infrastructure with minimum manpower and effort, the system was designed along three guidelines:

- Provide a production service to users as soon as possible
- Favour manageability and flexibility over performance
- Ensure interoperability with existing and upcoming infrastructures

This translated into a number of design and implementation choices that will be described in the following sections. We choose only stable and widely used tools and components, such as the OpenNebula cloud stack and the GlusterFS filesystem, and try to develop in-house as few pieces as possible. Furthermore, an agile development cycle was adopted, with resources given to the users as soon as possible and features and functionalities introduced only when they become needed by the applications.

2. The components

The middleware stack of choice is OpenNebula [4], an open-source software suite aimed at deploying, managing and monitoring Infrastructure-as-a-Service clusters widely used both in industry and in the research community. Even though most of the more recent Cloud projects use OpenStack [5], at the time we made the original choice the latter was judged not to be mature enough for a production environment. Furthermore, OpenNebula is arguably more suited for the task at hand (an advanced approach to data centre virtualization, with IaaS as a reference paradigm), whereas other stacks are more aimed at the full-fledged deployment of AWS-like infrastructures [6].

OpenNebula has many attractive features:

- it has a modular and easily customizable architecture, mainly based on shell and ruby scripts;
- it provides most of the needed features and functionalities, with the notable exception, as of version 3.8, of an integrated tool for network-on-demand, which prompted the developments described in section 5. ;
- it exposes, even if with different levels of maturity, industry-standard interfaces such as OCCI and EC2, along with the native OCA;
- it includes SunStone, a functional web application for most of the management tasks.

The versions used for the work described in this paper were 3.6 and later 3.8. Wide use is made of contextualization techniques, which allow us to maintain only a small number of very generic , very simple Operating System images, which are then set up and configured ("contextualized") only at boot time. We are currently using the standard contextualization tools natively provided by OpenNebula [7] and maintaining a number of shell scripts, but work is ongoing to adopt CloudInit [15], a more advanced tool recently adopted by many Cloud Computing projects.

Also for the backend storage (as opposed to *data* storage, which holds data to be accessed by the virtual workers, *backend* storage is used mainly to provide shared space to VMs and to distribute images across the physical hosts) a widely used tool was chosen, whose robustness and scalability has been proven by several very large deployments in the scientific and high-performance computing.

GlusterFS [8] is an open-source filesystem for network-attached storage currently developed and maintained by RedHat. It provides horizontal scalability by aggregating storage servers ("bricks") over the network into one large parallel file system with no master host: all synchronizations are peer-to-peer and clients access data directly from the node hosting it. While reasonably easy to set up in the simpler configurations, it is at the same time flexible enough to cater to different needs with a single tool; in particular, GlusterFS can mimic some RAID functionalities, like striping or mirroring, at the filesystem level so that a filesystem can be optimized for performance or reliability. This feature was exploited as described in section 4.

OpenNebula, while implementing tools for the definition and management of virtual networks at the cloud controller and hypervisor level, at the time lacked a tool for their implementation in the virtual infrastructure. We currently deploy Virtual Routers as small virtual machines based on OpenWRT [14], a lightweight Linux distribution designed for embedded systems; they provide internal and external connectivity to the virtual machines of a given application, alongside with tools for network configuration and management; details will be given in section 5.

3. A tale of two clusters

The components of most Scientific Computing systems can be very roughly divided into three categories: general services, worker nodes, and storage. General services are needed to orchestrate the computation, provide access to the cluster, manage and monitor the system or provide specific services for the relevant use case (a typical example is an LRMS head node); worker nodes are the number crunchers that perform the actual calculation (being it data analysis, simulation, numerical computation etc.); storage services provide access to input data to be analysed and, if needed, storage space for output. The latter are distinguished from the storage system needed by the Cloud Infrastructure to operate (e.g. for distributing the virtual machine images across hosts); storage services for data are beyond the scope of this paper and below we will only briefly describe one type of data storage we use.

"Servers" that provide general services and "Workers" that provide computational power have different requirements. In order to efficiently accommodate the two classes, the infrastructure

comprises two groups of hosts ("Clusters" in OpenNebula terminology), the "Services Cluster" and the "Worker Clusters"; "Server-class" virtual machines will be instantiated on the Services Cluster, and "Worker-class" ones on the Workers clusters.

The services are often critical for the functionality of the whole system, so they need resilient, redundant hardware and some form of High Availability may be desirable. Some services require inbound connectivity (at a bare minimum, for granting access to the system), whereas the local disk performance requirements may not be exceedingly tight. The hosts in this cluster are built on server-class hardware with redundant RAID disks and power supplies. They are dual-homed with interfaces in both the public network and the workers' private network (private and public IPs) with both in- and out-bound connectivity to provide outside access to private machines. Conversely, workers that do data analysis need a high-throughput access to data storage, usually don't need a public IP since data to be processed can be ether on local disk or on a local server reachable through the private network. Losing a Worker means only losing a small fraction of computational power, so they can use less redundant (and less expensive) hardware. The workers are typically dual-twin servers that share a redundant power supply among four motherboards, but have good network connection with the storage servers; the infrastructure currently includes about 40 such hosts.

Any generic application (a "virtual farm" in the following) on our infrastructure can be viewed as a number of virtual machines in the Workers cluster managed by a number of services that run in virtual machines in the Services cluster. At the bare minimum, the only service needed is a Virtual Router to provide network services and access from the outside to a user's virtual Worker Nodes.

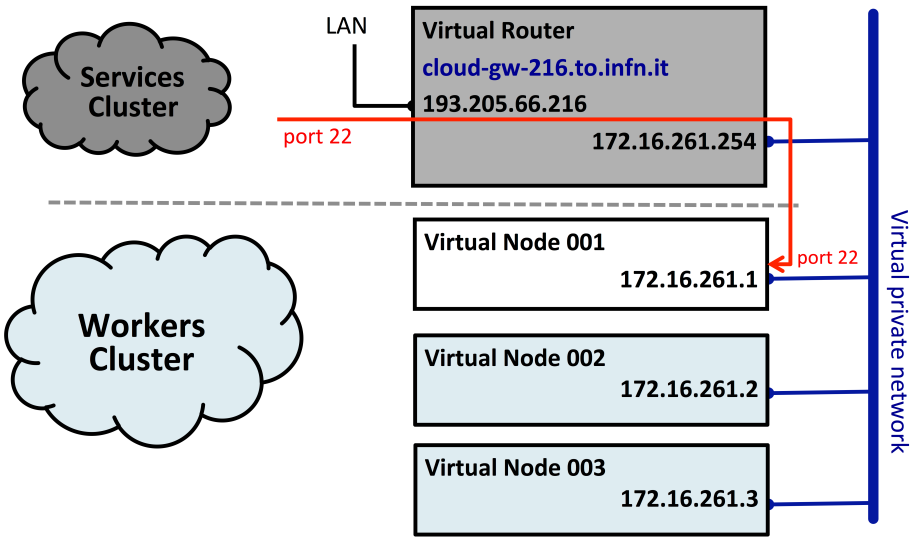

Figure 1. Example deployment of a Virtual Farm on the two clusters. VMs in the Workers Cluster are isolated from other VMs running on the infrastructure and connected to the external network through the Virtual Router. Higher-level services can be run on the Worker cluster head node (e.g. a batch queue manager) or, if they are more critical or need outbound connectivity, on a separate VM on the Services cluster.

The two clusters have been provided with different types of backend storage ("Datastores" in OpenNebula terminology) in order to optimize the performances and satisfy the above requirements, as described in section 3.

4. Storage backend architecture and GlusterFS

The backend storage is used to distribute virtual images, in our case either RAW or QCOW2, to the physical hosts. As mentioned above, the Services and the Workers clusters have different requirements, most of which are fulfilled by Gluster filesystems served by two fileservers equipped with redundant hardware (RAID disks and power supplies) and 10Gbps Ethernet interfaces. An external redundant FibreChannel controller provides the physical storage units.

The virtual images repository resides on a simple Gluster filesystem that currently comprises a single brick, but that can be scaled out by simply adding more bricks should more capacity be needed. All hosts mount this filesystem, from which the image needs to be copied by the relevant OpenNebula Transfer Manager to the datastore on which the running machine images will reside before it boots. Both the datastores and the copy mechanisms are different for the Services and Workers clusters.

As mentioned, services may be crucial to the smooth running of the whole site, so one wants them to be resilient, i.e. to be able to tolerate with little or no interruption of service the failure of a hardware or software component. To this aim, all hosts in the Services cluster share a Gluster filesystem on which the image is copied just before the virtual machine boots. The filesystem is replicated across the two fileservers to ensure availability in case of the failure of one, with the Gluster's self-healing feature enabled, and is shared across all physical hosts to allow for live migration of virtual machines from one host to another. Even if we currently don't implement automatic High Availability mechanisms, this at the very least allows us to e.g. put offline a physical host for maintenance without service interruption, at the cost of a latency in the startup of the virtual machine while the image is copied from the repository to the Shared Datastore; this is generally deemed tolerable since the restart of a service is a rare event. Also the small price to be paid in terms of performance from having a host accessing its running image over the network is justified by the higher stability of the system.

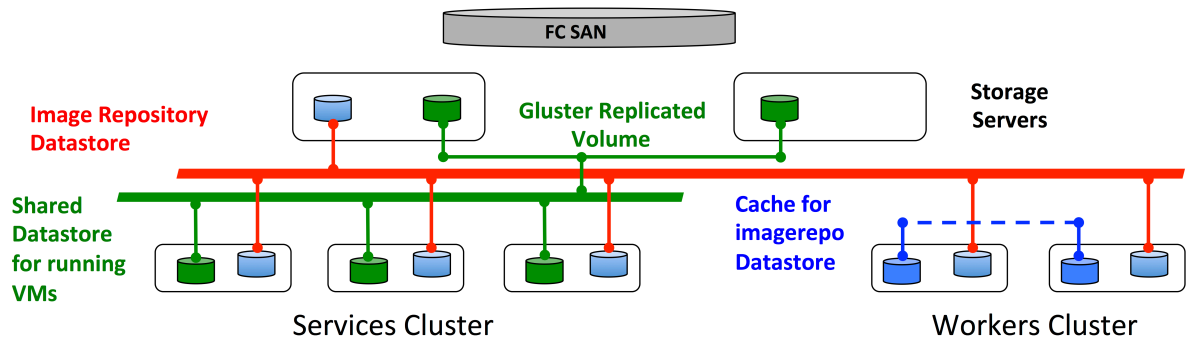

Figure 1. Architecture of the Backend Storage; Data Storage is not included.

On the other hand, dynamic allocation of resource entails the need for the fast startup of large numbers of virtual machines, which would be impossible if the images needed to be copied every time. Furthermore, the computing capacity of the site depends heavily on the performance of worker nodes, so network latency towards the running image is to be avoided. Thus, hosts in the Workers cluster share no filesystem; their running images are cached asynchronously on the local disk of the host by a custom torrent-like tool (built upon scpWave [9] and rsync [10]) and managed by a custom-

written Transfer Manager, so that they are always already there when a virtual machine needs to boot. Of course this implies that a little care is required in the management of the images in order not to saturate the local disks that comprise this Cached Datastore.

For the sake of completeness we just mention that also one of the applications running on the cloud infrastructure, the PROOF-based Virtual Analysis Facility [15], takes advantage of a performance-optimized 50TB Gluster filesystem also for data access.

5. Virtual Farm networking and the V-Router

As already mentioned, OpenNebula was lacking a tool to build and manage on-demand virtual networks to provide virtual farms with an isolated, flexible and secure network environment. In particular, for example, a tool was needed to provide access from the outside to private virtual machines.

In our infrastructure, all physical hosts share a common private network, while the ones in the Services cluster also have a second interface on a public network. Then, each application has its own private fully-featured class-C network built on top of the physical network. At level 2, these networks are isolated from each other using appropriate ebtables [11] rules defined on the hypervisor network bridge, to which obviously the users have no access.

At level 3, each private network (except the larger one for the WLCG Tier-2, which is managed directly on the core switch of the infrastructure) is managed by a Virtual Router: a small virtual machine (one physical CPU, 150 MB memory) running OpenWRT, a lightweight Linux distribution originally designed for embedded systems. OpenWRT provides tools for network configuration and management such as NAT and DHCP, firewalling and port forwarding along with a web interface for their configuration and monitoring. The Virtual Routers run on the services clusters, both for robustness and to allow them to be accessed from the outside.

In the standard configuration, port 22 on the public network of the Virtual Router is forwarded to port 22 on the private network of one of the nodes in the virtual farms, thus providing ssh access to manage the nodes; communication between nodes in the same virtual farms is open, whereas of course they have no way to access the private networks of other clusters.

6. Conclusions and outlook

Currently the Torino Cloud hosts two main applications: a fully-fledged WLCG Tier-2 site for the ALICE experiment, in which all Grid services except the Storage Elements run on the Services cluster and the Worker Nodes on the Workers cluster; and a PROOF-based Virtual Analysis Facility for the same experiment, which has been described in detail elsewhere [15]. This last application, based upon CernVM [11], PoD [13] and PROOF, is currently the one that better exploits the IaaS features of the infrastructure. Alongside those, a number of smaller use cases have been successfully tested and more are planned to join in the next future, including a Tier-2 centre for the BES-III experiment.

The infrastructure has now been in production for nearly two years; the tools of choice proved themselves satisfyingly stable and robust. This, together with the flexibility and manageability of the IaaS model, helped to reduce the management load on the Data Centre staff: as a trivial example, the frequent updates of the Grid middleware are now much faster and easier, as is reallocation of resources across different applications.

There is of course still much room for improvement, and we are planning several updates to the system, both to improve the stability of the infrastructure and to provide new functionalities.

For example, we are exploring the opportunities given by the CernVM “ecosystem”, in which the ongoing development of PROOF and the Virtual Analysis Facility is tightly entangled, to provide higher-level Platform-as-a-Service tools to experiments. We are also working on the Amazon EC2 interface of OpenNebula to ensure interoperability with other similar infrastructures and we plan to participate in upcoming projects aimed at developing higher-level federated cloud infrastructures. Moreover, we are moving the first steps towards providing self-service systems for advanced users to deploy their virtual application farm in as simple as possible way.

Acknowledgements
The present work is partially funded under contract 20108T4XTM of "Programmi di Ricerca Scientifica di Rilevante Interesse Nazionale" (Italy).
The authors want to acknowledge the support by the ALICE and BES-III communities, and in particular by their Italian components.

References
[1] http://wlcg.web.cern.ch/
[2] http://aliceinfo.cern.ch/
[3] http://aaf.cern.ch/
[4] http://www.opennebula.org
[5] http://www.openstack.org/
[6] See for example http://blog.opennebula.org/?p=4042
[7] http://opennebula.org/documentation:archives:rel3.8:context_overview
[8] http://www.gluster.org
[9] https://code.google.com/p/scp-wave/
[10] http://rsync.samba.org/
[11] http://ebtables.sourceforge.net/
[12] http://cernvm.cern.ch/portal/
[13] Malzacher P and Manafov A 2010 *J. Phys.: Conf. Ser.* **219** 072009
[14] https://openwrt.org/
[15] https://help.ubuntu.com/community/CloudInit
[16] Berzano D *et al.* 2012 *J. Phys.: Conf. Ser.* **368** 012019
[17] http://aws.amazon.com/ec2/