Dynamic Provisioning of Resources in a Hybrid Infrastructure

John White[*]
Helsinki Institute of Physics, Technology Program
E-mail: John.White@cern.ch

Salman Toor, Paula Eerola, Tomas Lindén, Oscar Kraemer
Helsinki Institute of Physics, CMS Program
E-mail: {salman.toor paula.eerola Tomas.Linden carl.kraemer}@helskinki.fi

Lirim Osmani, Sasu Tarkoma
Department of Computer Science, University of Helsinki
E-mail: {Lirim.Osmani sasu.tarkoma}@cs.helsinki.fi

We have built a hybrid Grid/Cloud cluster that allows CMS Grid users to submit jobs in a familiar manner. This cluster functions within the CMS production system in a basic form and we are now adding the more advanced features. In this paper we are presenting the results of using the EMI Argus and EES to instantiate new virtual machines within the OpenStack Cloud. The Argus service is used to render the authorization decisions based on X.509 credentials and EES interacts with the OpenStack Cloud infrastructure.

International Symposium on Grids and Clouds (ISGC) 2014,
23-28 March 2014
Academia Sinica, Taipei, Taiwan

[*]Speaker.

1. Introduction

The subject of data storage and analysis on Cloud infrastructures has gained importance in recent years. The High Energy Physics community is interested in the case for performing simulations and data analysis on public Cloud facilities. Currently the simulations and analysis are performed on a computing and data Grid. The software and experience of operating on a Grid needs to be adapted for running on Cloud facilities. The approach of harnessing Grid and Cloud ensures that the transition towards upcoming technologies will be steady and seamless.

This paper describes the state and current experience with a virtualized computing environment for CMS data analysis. In order to fulfill this task, a cluster has been constructed that is Cloud-based, Grid-enabled and runs with the Gluster file system. Our hybrid solution is built on Grid and Cloud software components incorporating dynamic provisioning of the computing resources through service endpoint URLs exposed for the end users and Cloud-aware systems by the Cloud service catalog. Also Argus and the Argus-EES (Authorization service and Execution Environment Service) by the European Middleware Initiative (EMI) [14], are deployed to authorize and initiate resource requests based on the incoming Grid jobs.

The Grid software that controls the execution of the physics jobs are components of the Advanced Resource Connector (ARC). This allows the end-users to submit the jobs with their preferred Grid or Cloud submission system and at the same time provides flexibility to maintain the infrastructure. The analysis software and libraries are installed via the CERN Virtual Machine File System (CernVM-FS), a common component currently on most Grid sites. The cluster performance is measured by running Site Availability Monitoring (SAM) jobs, CMS Monte Carlo simulation jobs, and CMS physics analysis jobs.

The cluster has been configured to provide a capacity compatible with a typical CMS data analysis site. In the first phase of the project, we have evaluated the strengths and weaknesses of this infrastructure with a semi-static approach in which the provisioning of worker node virtual machines was manual. Now we have enhanced the Argus-EES functionality to communicate with the OpenStack Cloud for dynamic provisioning of the resources. Our test results show that the adopted approach provides a scalable and resilient solution for managing resources without compromising on performance and high availability.

2. Grid and Cloud

In the scientific world, Grid computing has been a major driving force for providing large scale computational and data management capabilities. The aim is to join geographically-distributed resources to build a shared computational pool. The resources are under different administrative domains managed by independent organizations, which makes it difficult to provide uniform resource management. Globus toolkit [10] and the EMI provided components based on gLite [9], UNICORE [23], dCache [16] and ARC [15] are widely acceptable for structuring production-ready Grid deployments. The efforts and the role of the Grid model has been acknowledged in the Higgs discovery at CERN. While the Physics community has been the largest user of Grid systems, projects from life sciencess [22] [21], chemistry [20], and humanitiess [13] have also gained significant benefit from computational Grids.

Given that Cloud computing is the model chosen commercially in order to provide the most flexible, on-demand and cost-effective computing to customers, it is proving interesting for research communities to follow this model. Amazon [1], Google [11], OpenStack [12] and many other commercial and private Cloud are the big success stories of this model. There are proposals within the EU that future computing projects will not have funds allocated for establishing computing hardware and personnel, rather the computing resources will be "purchased" from Cloud providers. The Cloud providers in this case may be academic or commercial. In the case of the academic (or private) Cloud, the resources are generally owned and controlled by an institutional entity.

A detailed comparison of Grid and Cloud is presented in [19] . Both of these models allow better utilization of computational and storage resources but they are fundamentally different from each other, Cloud computing provides level of abstraction together with the notion of *lease* and the availability of resources are on-demand. Whereas the Grids are designed to *share* the resources amongst the collaborative partners and provides a massive batch processing system. It has also been realized that the underlying computational model also affects the design pattern of an application. Both of these models are still in use, hence it is required to extend infrastructures towards hybrid models that can fulfill the demands of service-oriented applications and at the same time allow the legacy applications to be executed in batch mode. The focus was on following key concepts while designing hybrid infrastructures:

- The infrastructure should combine the best of two models but avoid tight coupling with any of them.
- The interface to the hybrid infrastructure should be transparent enough so that both Grid and Cloud users can work in their familiar settings.
- The Grid and Cloud communities have different security and trust mechanisms. In order to bridge these mechanisms, it is important that the infrastructure allows users to connect with their trusted mechanism.

The model presented in this article relies on the above-mentioned concepts. One of the key features provided by the Cloud model is the dynamic provisioning of resources based on virtualization technology which enables applications to run in isolation within the required configuration settings. This feature is enabled for all the applications either coming from Grid or Cloud interfaces. The infrastructure is deployed using the OpenStack Cloud suite in order to avoid tight coupling using modular components that communicate via the standard HTTP(s) protocol. Due to the diversity in the security mechanism, the setup is equipped with a token-based authorization service for Cloud as well as a X.509 certificate-based security mechanism for the Grid users.

3. Mechanisms for Hybrid Infrastructure

The provisioning of resources in the infrastructure not only requires to identify the best available resource but also to check the user quota and their role within the project. For this purpose, Keystone identity service of OpenStack is used for the Cloud token-based authorization and authentication mechanism. In the Grid system, components for authorization and authenticate are available but require certain modifications/enhancements to build the connection with the available

Cloud suites. The presented approach of dynamic resource scaling in the hybrid infrastructure is closely connected with the available authorization services. Since we have extended the capabilities of Grid components, it is also important to understand the high-level concept behind the Grid system security.

The series of EU-funded Grid projects (DataGrid, EGEE-series and EMI) [6, 7] have provided a middleware set with integrated security at all levels. The security was mandated as a requirement from sites that run the middleware stack in order to provide user traceability and security against attacks from within the Grid itself. Grid sites also sign agreements, in the form of policy documents, that govern their behaviour in handling user data and jobs in terms of their privacy and security. Overall, this integrated security within the Grid middleware and obeyance of the policy documents gives Grid users the confidence their data and computing jobs are being handled appropriately.

The possibility that jobs injected into a Grid system may run on commercial Cloud resources gives rise to interesting questions to both the Grid users and resource providers:

- Can the data the user accesses or generates be stored on unknown commercial Cloud servers?
- Are algorithms, used in the job, proprietary or have they potential commercial value?
- Can a Grid resource provider be sure that the private or commercial Cloud resources are being used properly?

Based on the answers to these questions, the user may elect to avoid this computing model in the case of data and jobs being run on a commercial Cloud. For the case of data and jobs on a academic (private) Cloud then the same policies and security as assured in the Grid model should apply. For the last question, there needs to be a connection between the Grid security model and the Cloud computing stack(s).

In order to make a connection between Grid security and Cloud computing it was decided to investigate a connection between the EMI Argus authorization system and the OpenStack Cloud system.

4. System Components

4.1 OpenStack Cloud

OpenStack [12] is a collection of a number of distinct components initially produced by RackSpace and NASA. The project being currently in its ninth major release (Havana) consists of : OpensStack Compute (Nova), OpenStack Object Storage (Swift), OpenStack Block Storage (Cinder) OpenStack Image Service (Glance), Identity service (Keystone), OpenStack Networking (Neutron), OpenStack Orchestration (Heat) OpenStack Metering (Ceilometer) and Dashboard (Horizon) as individual components that operate autonomously but collectively can combine to create a very feature rich Cloud platform.

Each of the components provides a specific element of functionality and in typical deployments multiple machines are used running one or a set of the above-mentioned components that connect to each other via their open API's.

Our setup consist of a Cloud controller, network node, storage bricks and compute nodes. As the name implies, the Cloud controller is responsible for orchestration of the Cloud, with responsibilities for scheduling of instances, running the dashboard and the database store but also providing API endpoints to the public. The network node (Neutron) provides network services and manages the inbound and outbound network connectivity for the instances. The compute nodes are the simplest in their role as their main responsibility is to provide compute resources to the cluster and executing commands as instructed by the controller. Our storage solution consists of Cinder nodes providing block storage to running instances and GlusterFS bricks configured as single shared storage namespace for the virtual instances to boot into.

4.2 Advanced Resource Connector

The Advanced Resource Connector (ARC) is a set of open source Grid computing middleware components developed by NorduGrid [17] and distributed under the Apache License. It provides a common interface for submission of computational tasks to heterogeneous distributed computing systems and thus can enable Grid infrastructures of varying size and complexity. ARC includes data staging and caching functionality, developed in order to support data-intensive Grid computing.

The ARC components used in this test cluster are:

- A-REX, the ARC job execution service. The test and production CMS data analysis jobs are submitted to A-REX (with Condor back-end).
- JURA, job record publishing service for A-REX. This accounting service is forwarding job details to the SweGrid Accounting System (SGAS) used by NDGF [18] and CSC [8].
- GridFTP server for data staging.

5. Production Setup of DII-HEP Cloud

Figure 1 illustrates the DII-HEP production ready infrastructure deployed on the University of Helsinki Computer Science Department HPC “Ukko” cluster. The integration for the current modifications for dynamic resource provisioning is currently ongoing. The resources are homogeneous: Dell PowerEdge M610 servers with two quad core Intel Xeon E5540 processors, 32GB (8 x 4GB) 1066 MHz DDR3 ECC of RAM, 4 x 10GbE Broadcom 57718 network interfaces and 80GB SATA 7200rpm HDD. OpenStack Havana release is installed on top of Ubuntu 12.04 LTS.

The architecture consists of the following four components:

- Physical Storage: System storage consists of 10 servers. Each of the servers has a 512 GB LUN attached from a storage array over the “Fibre Channel over Ethernet” protocol. 4 of them are used to build the Cloud infrastructure, whereas rest of the 6 servers have been used inside the Cloud, dedicated to applications and system configurations.
- Gluster File System: On top of the physical storage, Gluster File System has been deployed. Based on the storage Bricks, GlusterFS promises to provide efficient, scalable and fault-tolerant solution for peta-scale data management.

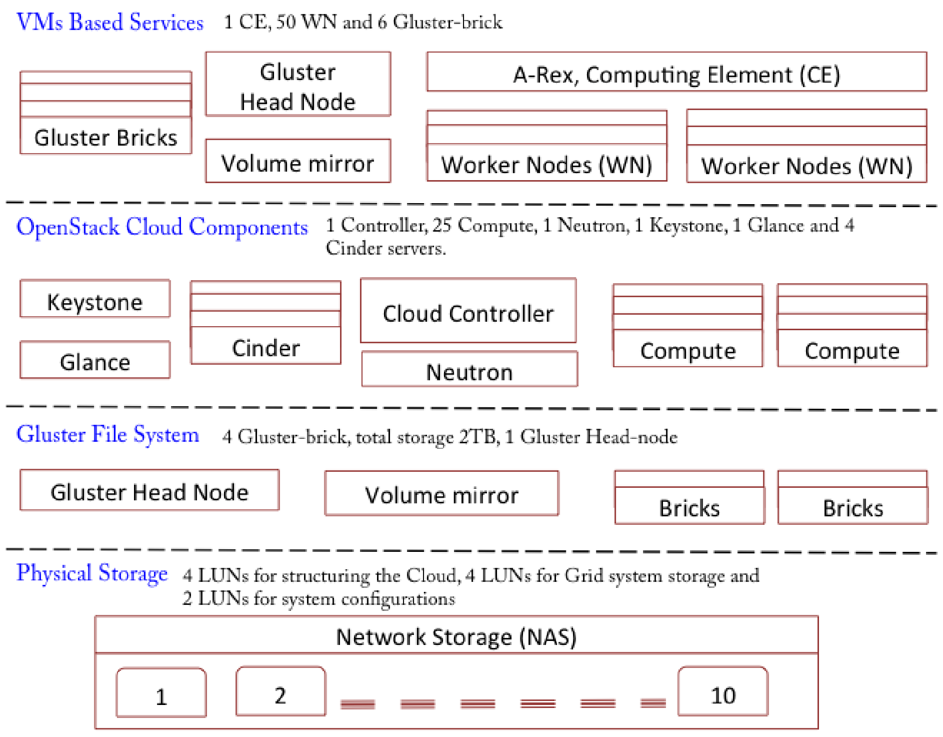
VMs Based Services
1 CE, 50 WN and 6 Gluster-brick
Gluster Bricks
Gluster Head Node
Volume mirror
A-Rex, Computing Element (CE)
Worker Nodes (WN)
Worker Nodes (WN)
OpenStack Cloud Components
1 Controller, 25 Compute, 1 Neutron, 1 Keystone, 1 Glance and 4 Cinder servers.
Keystone
Glance
Cinder
Cloud Controller
Neutron
Compute
Compute
Gluster File System
4 Gluster-brick, total storage 2TB, 1 Gluster Head-node
Gluster Head Node
Volume mirror
Bricks
Bricks
Physical Storage
4 LUNs for structuring the Cloud, 4 LUNs for Grid system storage and 2 LUNs for system configurations
Network Storage (NAS)
1
2
10

Figure 1: Overall system architecture based on OpenStack and GlusterFS.

- OpenStack Cloud Components: This section contains all the OpenStack components used to build the infrastructure. The component-based model of OpenStack Cloud allows to build resilient services in the system.

- Virtual Machine-based Services: DII-HEP enables a Virtual Machine(VM)-based execution environment for Grid jobs. All the necessary Grid middleware components together with GlusterFS based shared storage pool are running on number of different VMs. The selection of software components such as those from ARC and CernVM-FS make the site seamless to end users familiar with Grid systems and also ease the interface to Grid accounting and monitoring systems.

Recently in [24], we have reported the scalability and performance evaluation based on DII-HEP setup. The presented Cloud setup allows flexibility and elasticity in system deployment and management. The use of the Grid middleware components and monitoring based on Grid tools allows seamless integration of this system with other Grid sites. Also Grid users are able to submit their jobs using legacy Grid interfaces.

6. Plugin Extension Based on Argus and EES

We have developed a plugin for the Argus Execution Environment Service (EES) that starts (and stops) virtual machines within the OpenStack cluster.

6.1 EMI Argus

The Argus Authorization Service [4] renders consistent authorization decisions for distributed services (e.g., user interfaces, portals, computing elements, storage elements). The service is based on the XACML standard, and uses authorization policies to determine if a user is allowed or denied to perform a certain action on a particular service.

The Argus Authorization Service is composed of three main components:

Policy Administration Point (PAP)

- Provides the tools to author authorization policies, organizes them in the local repository and configure policy distribution among remote PAPs.

Policy Decision Point (PDP)

- Implements the authorization engine, and is responsible for the evaluation of the authorization requests against the XACML policies retrieved from the PAP.

Policy Enforcement Point client (PEP client)

- Lightweight PEP client libraries are also provided to ease the integration and interoperability with other EMI services or components.

Policy Enforcement Point Server (PEP Server)

- Handles the authorization requests received from the PEP client, and ensures the integrity and consistency. Configurable policy information points (PIP) can transform or complete the incoming requests. Obligation handlers (OH) can be applied to the resulting response.

Figure 2 gives a schematic illustration of the relationship between the key Argus components.

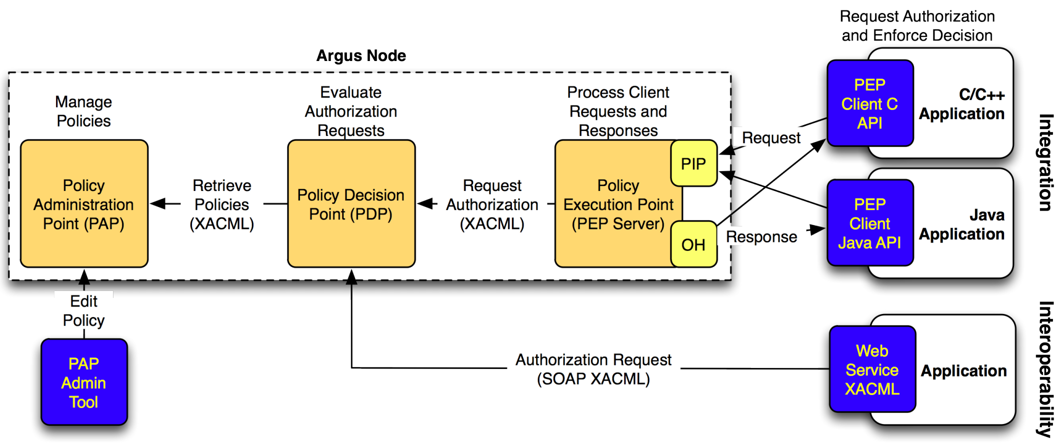
Request Authorization and Enforce Decision
Argus Node
Manage Policies
Evaluate Authorization Requests
Process Client Requests and Responses
Policy Administration Point (PAP)
Retrieve Policies (XACML)
Policy Decision Point (PDP)
Request Authorization (XACML)
Policy Execution Point (PEP Server)
PIP
OH
Request
Response
PEP Client C API
C/C++ Application
PEP Client Java API
Java Application
Integration
Edit Policy
PAP Admin Tool
Authorization Request (SOAP XACML)
Web Service XACML
Application
Interoperability

Figure 2: EMI Argus components.

A further component is needed to act on the OpenStack installation based on the decisions reached by the Argus PDP. This is a plugin that has been written for the Argus Execution Environment Service (EES).

6.2 EMI Execution Environment Service

The Execution Environment Service (EES) [5], is the site-local mapping component of Argus. It converts Grid-wide or logical obligations into site-specific execution environments, supported by any attribute the job carries from the user or the VO. Examples may be the assignment of a site-local unix uid out of a pool based on FQAN attributes given by the VO; or configuring a machine image to be executed in a virtual machine hosting environment on a worker node. The EES is a pluggable, configurable, standalone service responding to requests from a Policy Enforcement Point (PEP) which have been augmented with information from a Policy Decision Point (PDP).

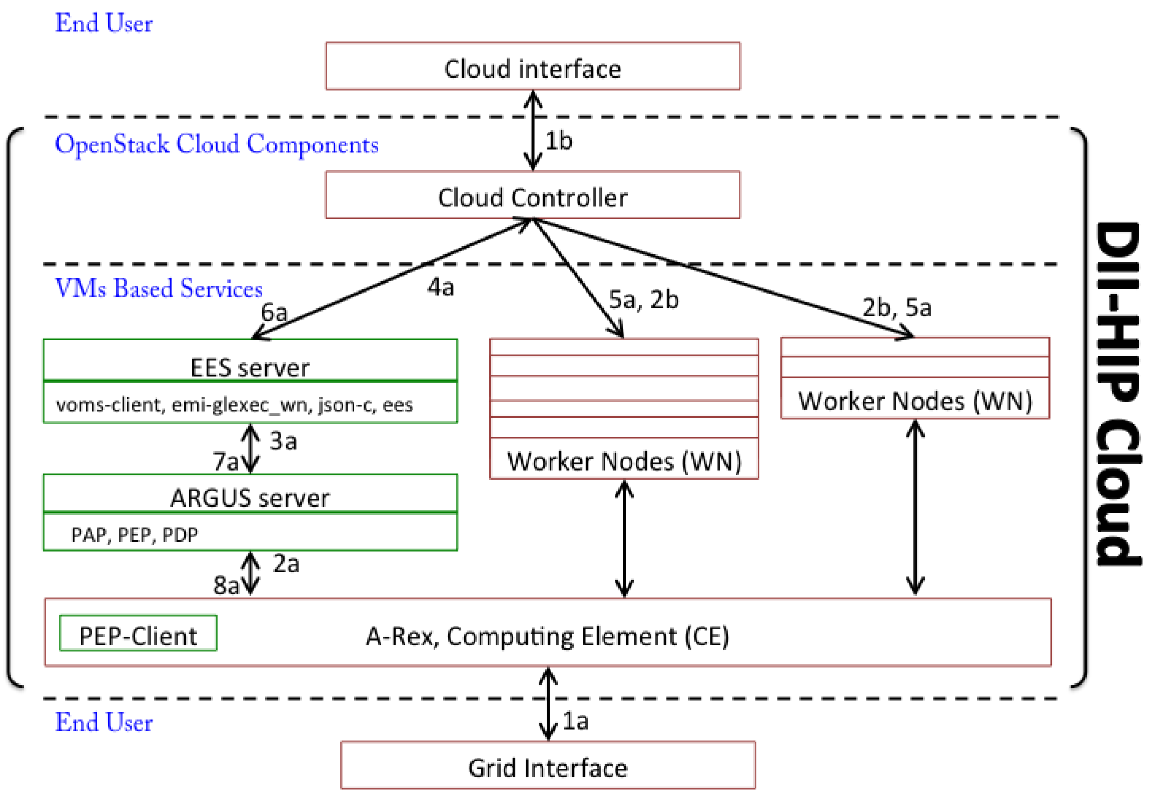

Figure 3: The role of EES with Argus, in the context of VM scheduling.

6.3 EES OpenStack plugin

In order to control the start/stop of VMs in the OpenStack installation, a new plugin for the EES has been written. This EES OpenStack plugin communicates with the OpenStack installation using the provided OpenStack JSON API. The EES plugin handler is written in C and so the standard JSON-c library was used to parse the OpenStack JSON responses. In order to keep the plugin as standard as possible the `curl` library [2] was used to communicate with OpenStack (in same manner as the OpenStack nova client does).

Rather than present a textual description of the sequence of events that start an OpenStack VM, Figure 4 gives the sequence diagram:

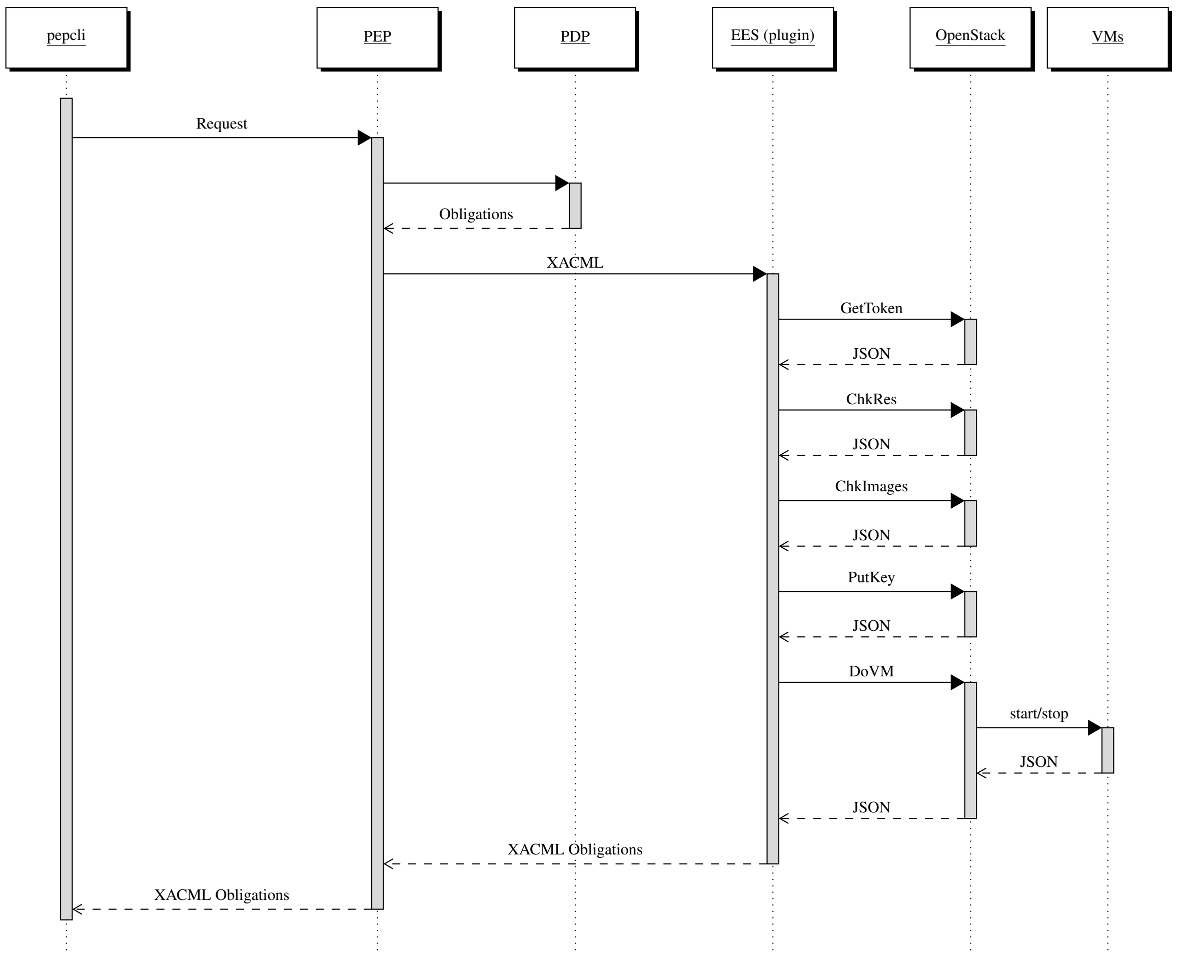

Figure 4: Sequence diagram from `pepcli` to OpenStack VM.

The new work that was needed to start and stop VMs in the OpenStack installation was the EES plugin coding. As mentioned above, the EES OpenStack plugin is written in C for compatibility with the EES framework. The format of the plugin follow the guidelines given in [3] and use the `curl` library API to communicate with the OpenStack controller node. The JSON responses from OpenStack are parsed using the json-c library. The XACML obligations are handled by the EMI saml2-xacml2 library.

7. Plugin Evaluation

The test setup used for the Argus-EES to OpenStack interaction is very simple, as can be seen in Figure 3. For convenience, the Argus service and EES are deployed on two separate VMs, `argus-1` and `ees`, within the OpenStack Cloud VM pool. The test client, the Argus-supplied `pepcli`, is also deployed on `argus-1`. This is not a necessary requirement as `pepcli` may be deployed on any machine as long as the network and firewall allows a connection to the Argus service port. The pepcli contacts the Argus PEP server, described in Section 6.1. In turn, the PEP receives the authorization result from the PDP and contacts the EES. The EES, on ees, runs a series of plugins including the OpenStack plugin. The OpenStack plugin handles the curl calls that pass and receive the JSON strings to and from the OpenStack controller node. It is here that the OpenStack VMs are started (or stopped). The OpenStack controller node is, of course, run on a physical machine outside of the VM pool.

The test scenario involves starting multiple OpenStack VMs using pepcli . A script starts the VMs and reports the time from VM start to the first successful ping response. According to this simple test, all VMs booted successfully with no failures. The average time for first successful ping is 37s. In order to avoid network latencies, the pepcli is on the same network as rest of the components.

From this experiment shows following three points can be inferred:

- The EES OpenStack plugin is stable and works as expected. The success rate is 100%.
- The mechanism of authorization creates minimal overhead. This can be seen by the average client response time.
- The use of curl libraries make the request of VM booting comparable to the native API calls.

No. of Instances	Average Client Response (s)	Success Rate
1	1.17	100%
5	1.45	100%
10	1.32	100%
20	1.29	100%

8. Conclusion

This paper gives our first results in dynamically provisioning VMs within our hybrid Grid/OpenStack Cloud infrastructure. This is the first attempt at matching the authorization capabilities of the EMI Argus to the Cloud operations of OpenStack through the use of the EES.

Currently, the EES plugin has performed in a stable manner and no failures have been observed. In the future the aim will be to further evaluate the plugin performance. Argus, EES and the plugin will be integrated into production setup to start and stop the VMs for Grid jobs as they arrive and finish respectively.

References

[1] Amazon web services. http://aws.amazon.com.

[2] curl. http://curl.haxx.se/.

[3] EES plugin guidelines. http://www.nikhef.nl/grid/ndpf/files/EMI_1_SAC_documentation/ees-Task.pdf.

[4] EMI Argus. http://www.eu-emi.eu/kebnekaise-products/-/asset_publisher/4BKc/content/argus.

[5] EMI Execution Environment Service.
https://twiki.cern.ch/twiki/bin/view/EMI/ARGUS_EESv0_0_10Details.

[6] EMI Project. http://www.eu-emi.eu/.

[7] EU Datagrid. http://eu-datagrid.web.cern.ch/eu-datagrid/.

[8] Finnish Centre for Scientific Computing. http://www.csc.fi.

[9] gLite middleware. http://glite.web.cern.ch/glite/.

[10] Globus. http://toolkit.globus.org/toolkit/.

[11] Google Cloud. https://cloud.google.com.

[12] Openstack, Open source software for building cloud infrastructure. http://openstack.org/.

[13] The Arts and Humanities Data Service (AHDS). http://www.ahds.ac.uk/index.htm.

[14] C. Aiftimiei, A. Aimar, A. Ceccanti, M. Cecchi, A. Di Meglio, F. Estrella, P. Fuhrmam, E. Giorgio, B. Konya, L. Field, J.K. Nilsen, M. Riedel, and J. White. Towards next generations of software for distributed infrastructures: The european middleware initiative. In *E-Science (e-Science), 2012 IEEE 8th International Conference on*, pages 1–10, Oct 2012.

[15] Owen Appleton, David Cameron, Jozef Cernák, Péter Dóbé, Mattias Ellert, Thomas Frågåt, Michael Grønager, Daniel Johansson, Johan Jönemo, Josva Kleist, et al. The next-generation arc middleware. *annals of telecommunications-annales des télécommunications*, 65(11-12):771–776, 2010.

[16] G Behrmann, P Fuhrmann, M Gronager, and J Kleist. A distributed storage system with dcache. *Journal of Physics: Conference Series*, 119(6):062014 (10pp), 2008.

[17] P. Eerola et al. The NorduGrid production Grid infrastructure, status and plans. In *Proceedings of Fourth IEEE International Workshop on Grid Computing*, pages 158–165, 2003.

[18] Kleist J. Fischer L. Grønager M. and Smirnova O.˙ A Distributed Tier-1 for WLCG. In *Journal of Physics: Conference Seriesm 119*. IOP Publishing, 2008.

[19] I. Foster, Yong Zhao, I. Raicu, and Shiyong Lu. Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008. GCE '08*, pages 1–10, Nov 2008.

[20] Paolo Giannozzi, Stefano Baroni, Nicola Bonini, Matteo Calandra, Roberto Car, Carlo Cavazzoni, Davide Ceresoli, Guido L Chiarotti, Matteo Cococcioni, Ismaila Dabo, Andrea Dal Corso, Stefano de Gironcoli, Stefano Fabris, Guido Fratesi, Ralph Gebauer, Uwe Gerstmann, Christos Gougoussis, Anton Kokalj, Michele Lazzeri, Layla Martin-Samos, Nicola Marzari, Francesco Mauri, Riccardo Mazzarello, Stefano Paolini, Alfredo Pasquarello, Lorenzo Paulatto, Carlo Sbraccia, Sandro Scandolo, Gabriele Sclauzero, Ari P Seitsonen, Alexander Smogunov, Paolo Umari, and Renata M Wentzcovitch. Quantum espresso: a modular and open-source software project for quantum simulations of materials. *Journal of Physics: Condensed Matter*, 21(39):395502, 2009.

[21] D.B. Keator, J.S. Grethe, D. Marcus, B. Ozyurt, S. Gadde, S. Murphy, S. Pieper, D. Greve, R. Notestine, H. J. Bockholt, and P. Papadopoulos. A national human neuroimaging collaboratory enabled by the biomedical informatics research network (birn). *Information Technology in Biomedicine, IEEE Transactions on*, 12(2):162–172, March 2008.

[22] Natalia Maltsev, Elizabeth Glass, Dinanath Sulakhe, Alexis Rodriguez, Mustafa H Syed, Tanuja Bompada, Yi Zhang, and Mark D'Souza. Puma2 grid-based high-throughput analysis of genomes and metabolic pathways. *Nucleic Acids Research*, 34(suppl 1):D369–D372, 2006.

[23] A. Streit, D. Erwin, Th. Lippert, D. Mallmann, R. Menday, M. Rambadt, M. Riedel, M. Romberg, B. Schuller, and Ph. Wieder. Unicore — from project results to production grids. In Lucio Grandinetti, editor, *Grid Computing The New Frontier of High Performance Computing*, volume 14 of *Advances in Parallel Computing*, pages 357 – 376. North-Holland, 2005.

[24] S. Toor, L. Osmani, P. Eerola, O. Kraemer, T. Lindén, S. Tarkoma, and J. White. A scalable infrastructure for cms data analysis based on openstack cloud and gluster file system. In *Journal of Physics: Conference Series*. IOP Publishing, in press.