ATLAS user analysis on private cloud resources at GoeGrid

This content has been downloaded from IOPscience. Please scroll down to see the full text.
(http://iopscience.iop.org/1742-6596/664/2/022020)

View the table of contents for this issue, or go to the journal homepage for more

Download details:
IP Address: 188.184.3.52
This content was downloaded on 06/01/2016 at 16:08

Please note that terms and conditions apply.
ATLAS user analysis on private cloud resources at GoeGrid

F Glaser¹, J Nadal Serrano², J Grabowski¹ and A Quadt²

¹Institute of Computer Science, University of Göttingen, Germany
²II. Institute of Physics, University of Göttingen, Germany

E-mail: fabian.glaser@cs.uni-goettingen.de, jordi.nadal@cern.ch, jens.grabowski@cs.uni-goettingen.de, arnulf.quadt@cern.ch

Abstract. User analysis job demands can exceed available computing resources, especially before major conferences. ATLAS physics results can potentially be slowed down due to the lack of resources. For these reasons, cloud research and development activities are now included in the skeleton of the ATLAS computing model, which has been extended by using resources from commercial and private cloud providers to satisfy the demands. However, most of these activities are focused on Monte-Carlo production jobs, extending the resources at Tier-2. To evaluate the suitability of the cloud-computing model for user analysis jobs, we developed a framework to launch an ATLAS user analysis cluster in a cloud infrastructure on demand and evaluated two solutions. The first solution is entirely integrated in the Grid infrastructure by using the same mechanism, which is already in use at Tier-2: A designated Panda-Queue is monitored and additional worker nodes are launched in a cloud environment and assigned to a corresponding HTCondor queue according to the demand. Thereby, the use of cloud resources is completely transparent to the user. However, using this approach, submitted user analysis jobs can still suffer from a certain delay introduced by waiting time in the queue and the deployed infrastructure lacks customizability. Therefore, our second solution offers the possibility to easily deploy a totally private, customizable analysis cluster on private cloud resources belonging to the university.

1. Introduction

Several sites involved in the ATLAS computing model already extend their capacity by using additional computing resources from cloud providers. Thereby, the most commonly used approach is to extend the capacity of the local batch farm by launching additional worker nodes in a cloud environment. The CloudScheduler [1] allows to monitor a designated HTCondor [2] queue and manage additional worker nodes according to the demand in a cloud. Thereby, it is able to launch and terminate instances on several different cloud frameworks including Amazon EC2 [3] and OpenStack [4]. A CloudScheduler is deployed on the same machine as the HTCondor scheduler. It monitors its job queue and launches worker nodes for incoming jobs on a preconfigured cloud site on demand. These worker nodes automatically register with HTCondor and are then able to receive and execute the designated jobs. When the load on the cluster decreases, the CloudScheduler terminates the launched worker nodes. By exploiting HTCondor’s ClassAd mechanism, it allows the user to configure characteristics of the virtual machine to be launched, e.g. the machine flavor, the required amount of RAM, or the image file to launch the virtual machine from. While the solution introduced...
above allows to easily extend Grid and cluster resources with worker nodes launched in cloud environments, it is limited to the configuration of additional computing nodes and lacks the possibility to manage more sophisticated infrastructures. Today’s Infrastructure-as-a-Service (IaaS) frameworks, like OpenStack or Amazon’s EC2 allow to manage not only virtual machines, but also virtual networks, additional storage and offer additional higher level services that simplify the management and scaling of the deployed infrastructure. To fully leverage the potential of cloud computing for the high energy physics community, these higher level services and additional features need to be exploited.

We have developed a framework that uses a stack of open-source tools and standards to be able to exploit additional cloud features and simplify the manageability of private analysis clusters in the cloud. Thereby we utilize a template language, called Topology Orchestration and Specification for Cloud Applications (TOSCA) that allows one to define the cluster architecture in an abstract and reusable manner.

2. The Topology and Orchestration Specification for Cloud Applications

TOSCA is an evolving standard developed by the Organization for the Advancement of Structured Information Standards (OASIS). It aims to standardize a template language to describe cloud applications in a portable and reusable manner. The template language is similar to the languages used by Amazon’s CloudFormation and OpenStack Heat. Originally, TOSCA was defined in XML, but a simplified YAML profile is also available.

The basic components of TOSCA are depicted in Figure 1. A Service Template consists of a Topology Template and a Plan. The Topology Template defines the topology of an application supposed to be executed in the cloud, while Plans define how the application is managed and deployed. Topology Templates consists of Node Templates that define e.g. the virtual machines or application components and Relationship Templates that encode the relationships between them, e.g. a certain application component should be deployed on a certain virtual machine. Node Templates and Relationship Templates are of certain Node Types or Relationship Types respectively. These types define Properties, e.g. the IP address of a virtual machine and Interfaces, which define the operations that can be executed on the component, e.g. the termination of a certain application component.
We use TOSCA to define the topology and the software configuration of our private analysis cluster. To be able deploy TOSCA Topology Templates a cloud orchestration tool and a configuration management tool is needed to enforce the defined software configuration. The framework and the corresponding tool-stack is introduced in the next section.

3. The framework

Figure 2 depicts the different components of our framework on the left and the corresponding tools that are utilized on the right. On the topmost layer, there is the application that is supposed to be deployed in a cloud environment. In our case, we deploy an HTCondor cluster with ROOT [8] for user data analysis. This application is modelled with help of TOSCA, which allows us to easily adapt the infrastructure. The application models are then managed and executed by a cloud orchestration tool. The cloud orchestration tool is hereby responsible for creating and managing the virtual infrastructure. In our case, we decided to use Cloudify [9], which offers an easy-to-use web interface that allows to manage TOSCA YAML profile templates, initialize these templates to create concrete deployments by configuring concrete parameter values for deployments and roll-out the infrastructure in the cloud. Besides cloud orchestration, a configuration management tool is utilized to enforce the correct software configuration on the instances launched in the cloud. Cloudify is able to work with different configuration management tools. In our infrastructure, we decided to use Ansible [10]. To be able to connect to the launched instances in the cloud, a minimum configuration, such as specific user accounts and user credentials are needed. Enabling these configurations is called contextualization. The contextualization tool, we use in our infrastructure is called Cloud-Init [11]. The resources themselves are deployed on an IaaS cloud at the lowest level. In our infrastructure we utilize an OpenStack cloud. The minimal configuration of the virtual machines, such as the operating system is encapsulated in predefined images. We decided to utilize the CernVM base image [12] to boot the virtual machines from. Through the use of the CernVM File System (CernVM-FS), which is already preconfigured on the CernVM the user is able to load preconfigured analysis software in the cluster environment easily. The last component in the framework is the Data Management. It is responsible for providing access to the data to be processed by the cloud resources. We use the Federated ATLAS storage systems using XRootD (FAX) [13] to process data stored in the Grid on the cloud clusters.
4. Reusable cluster templates

The architecture of the deployed clusters is shown in Figure 3. A master node is configured to run the HTCondor scheduler. It also serves as a gateway to the cluster deployed in the cloud by getting a public IP address assigned. Currently, a fixed number of worker nodes is deployed that connect to the HTCondor scheduler. During the time of writing, Cloudify did not support the automatic scaling features of TOSCA, but these features are planned for future versions and will be incorporated in the framework when they are available. Additionally, the monitoring tool Ganglia [14] is deployed for monitoring. All nodes in the cluster are launched from a CernVM image. The name resolution inside the cluster is done with help of a local Avahi [15] installation.

The Grid certificate of the user launching the cluster are copied to the cluster automatically, when the cluster is launched such that he is able to work with FAX directly when the cluster is ready.

Listing 1 exemplifies the usage of the TOSCA YAML profile by listing parts of the master node definition. The master node is of type `cloudify.openstack.nodes.Server` which represents a basic virtual machine in an OpenStack cloud. This type defines several properties which can be set in the template definition. The internal function `get_input` can be used for telling Cloudify to ask for user input for a specific value. Here, the user is asked to give the ID of an image he wants to start the virtual machine for the master node from, the virtual machine type (flavor) and userdata script that is than executed by the Contextualization tool. Two relationship templates are defined: The first connects the virtual machine to a public IP address in the cloud (floating ip), while the second connects the virtual machine to a `security group`, a component that encapsulates the firewall configuration.
Listing 1. Cluster master node definition in a TOSCA Topology Template.

```yaml
crnode_templates:
cluster_master:
type: cloudify.openstack.nodes.Server
properties:
  ...
  server:
    image: { get_input: image }
    flavor: { get_input: flavor }
    name: cluster_master
  userdata:
    type: http
    url: { get_input: user_data }
relationships:
- target: master_floatingip
  type: cloudify.openstack.server_connected_to_floating_ip
- target: cluster_security_group
  type: cloudify.openstack.server_connected_to_security_group
```

5. User workflow

Figure 4 shows the overall workflow of users interacting with the Cloudify web interface to launch a private analysis cluster. In the first step, they select from a predefined list of cluster templates the one that suits their needs. In the following, they are able to customize the deployment with predefined parameters. Hereby, the user is able to configure the image to launch the worker nodes from, the virtual machine type to use and additionally also his Grid credentials for data access. After the configuration, the cluster is ready to be deployed. In the deployment process, Cloudify creates the virtual infrastructure on top of the IaaS cloud and Ansible enforces the correct configuration. After the deployment process the users can use the resources by remotely connecting to the cluster with help of the Secure Shell (SSH). They can hereby access and write data from and to the Grid by using their preconfigured Grid credentials and use any of the software packages enabled through CernVM-FS. During their work on the cluster, they are able to monitor the load through Ganglia. After they have done their analysis, they can terminate the cluster resources via the Cloudify web interface.

![User workflow diagram](image)

**Figure 4. User workflow.**

6. Computing Environment

The Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen (GWDG) serves as a local IT service provider for the university and the institutes of the Max Planck Society in Göttingen. With the private GWDG-compute cloud, it offers IaaS to provide highly customizable, on-demand computing resources to the scientists of the designated institutions. In cooperation with the biomedicine, the TextGrid [16], the department of physics, and the high energy physics in Göttingen, the GWDG also hosts the GoeGrid [17] Grid resource center that serves as a Tier-2 Grid site in scope of the WLCG. Due to the close proximity, it offers the chance for high speed data access from the cloud site to the data stored at GoeGrid.
7. Summary and Outlook
In this paper, we have presented a novel approach to deploy private ATLAS user analysis clusters in an IaaS cloud using open-source tools that are widely adopted in the industry. By taking advantage of the TOSCA template language for cloud orchestration, we simplify the customization of the whole cluster infrastructure and enable its reusability. The introduced framework has been prototypically implemented and is ready to be used to launch user analysis clusters. In the future, we plan to extend the developed solution to be able to scale the deployed clusters automatically according to the demand.

Acknowledgements
The work of Fabian Glaser is partially funded by the Joint Centre of Simulation Technology (SWZ) [18] of the University of Göttingen and the Technical University of Clausthal (project 11.4.1).

References
[12] CernVM cernvm.cern.ch/ last accessed 05/14/2015
[16] TextGrid https://www.textgrid.de/ last accessed 05/14/2015
[18] SWZ http://www.simzentrum.de/en/ last accessed 05/14/2015