Using Puppet to contextualize computing resources for ATLAS analysis on Google Compute Engine

Henrik Öhman\textsuperscript{a}, Sergey Panitkin\textsuperscript{b}, and Valerie Hendrix\textsuperscript{c} on behalf of the ATLAS Collaboration

\textsuperscript{a}Uppsala University (SE), \textsuperscript{b}Brookhaven National Laboratory (US), \textsuperscript{c}Lawrence Berkeley National Lab. (US)

E-mail: \textsuperscript{a}henrik.ohman@physics.uu.se, \textsuperscript{b}panitkin@bnl.gov, \textsuperscript{c}vchendrix@lbl.gov

Abstract.

With the advent of commercial as well as institutional and national clouds, new opportunities for on-demand computing resources for the HEP community become available. The new cloud technologies also come with new challenges, and one such is the contextualization of computing resources with regard to requirements of the user and his experiment. In particular on Google’s new cloud platform Google Compute Engine (GCE) upload of user’s virtual machine images is not possible. This precludes application of ready to use technologies like CernVM and forces users to build and contextualize their own VM images from scratch. We investigate the use of Puppet to facilitate contextualization of cloud resources on GCE, with particular regard to ease of configuration and dynamic resource scaling.

1. Introduction

Commercial and institutional and national clouds have been providing Infrastructure-as-a-Service (IaaS) as computing resources for the High Energy Physics (HEP) community for several years. A common way to configure these resources is to create custom machine images which have all the necessary software pre-installed and pre-configured. This can be a time consuming task with possibly high turnaround times for configuration and diagnostics. An alternative approach to creating such custom machine images is to use software for system configuration which enables configuration of the machine from a bare OS installation.

ATLAS analysis is the task of analyzing data samples collected with the ATLAS experiment \cite{1}. It involves running distributed analysis software over data and simulated samples using computing resources affiliated to the ATLAS experiment.

In Section 2 the components of a typical computer cluster for ATLAS analysis is described. Section 3 describes the contextualization procedure and the source code developed for the present work. In Section 4 the results of the study is presented, and results of a related study are cited. The paper is concluded in Section 5.

The descriptions in Section 2 should be viewed as related work, and is not a contribution to the this paper.
1.1. Computing resources for ATLAS analysis
The majority of computing resources for ATLAS analysis are distributed throughout the world and joined together via the Worldwide LHC Computing Grid (WLCG), the Open Science Grid (OSG), and NorduGrid. This wide distribution of resources create a necessarily heterogeneous flora of systems and system configurations. Indeed, different computing sites use different hardware, different batch scheduling software, different storage configurations, and so on. At the same time all computing elements must provide a homogeneous environment for experiment software, such that any processing and analysis is reproducible and that the generated results are reliable. Both the heterogeneous and the homogeneous parts of the software for ATLAS analysis clusters are discussed below.

1.2. Puppet
Puppet [2] is software that automates many tasks that system administrators perform. Puppet is operated by declarative manifests that define a desired system state. The desired state is then enforced on the system, protecting it from drifts in the configuration. Puppet can also simulate changes before enforcing them, and report what changes have been made.

1.3. Google Compute Engine (GCE)
With Google’s new cloud for IaaS, Google Compute Engine (GCE) [3], uploading of custom machine images is not possible. GCE has its own proprietary tools for customizing images that are not usable to create cloud independent machine configurations. In this context Puppet is particularly useful, enabling the creation of truly cloud independent, and indeed even infrastructure independent machine configurations.

2. Elements of an ATLAS analysis cluster
An ATLAS analysis cluster is a computing cluster connected to the PanDA (Production and Distributed Analysis) [4] system for ATLAS. For this purpose several functions are required. The cluster must be able to run the pilot jobs [5] that retrieve the payloads from the PanDA queue; it must have a local batch system that can manage the individual jobs; it must have the right connections to distributed storage; and it must be able to access CERN and ATLAS analysis and processing software. Below we describe the components and setup of the cluster used in this study.

2.1. AutoPyFactory
AutoPyFactory [6] is a modular, pluggable system able to collect jobs from different job queues in different ways and submit them to different types of batch systems. It works by submitting empty wrapper jobs to the batch cluster which download and run the pilot code. The pilot jobs in turn request a job from the specified PanDA queue. If a job is found its payload is retrieved and executed. The AutoPyFactory submits jobs to the batch system manager, and on our cluster it runs on the manager node.

2.2. HTCondor
HTCondor [7] is a job management system specially developed for high throughput computing. It offers the regular functions of batch system software, like queue management, job scheduling and prioritizing, and management and monitoring of computing resources. It also provides capabilities to match jobs to resources, and facilities to tie together several smaller clusters situated in different administrative locations. The HTCondor setup in our cluster consists of one manager node which runs the job submission and job brokering services, and one or more worker nodes which run individual jobs.
2.3. XRootD
The XRootD system [8] consists of several services to enable low latency access to large amounts of data. It can at the same time provide access to globally stored data (through streaming or caching) and locally stored data. The configuration possibilities are too numerous to be given justice here, so only the setup chosen for our cluster is described. In this setup each worker node acts as an XRootD data server which provides access to files stored on the node and storage for incoming data. The Cluster Management Service creates a storage cluster of the XRootD data servers.

To manage the storage cluster, the manager node (running the HTCondor manager and the AutoPyFactory service) acts as an XRootD redirector which redirects requests for files or storage to worker nodes where it is available. The XRootD service on the manager node is in turn connected to a redirector outside the cluster which provides access to ATLAS data and simulated samples. In our setup this global redirector is a member of the Federated ATLAS XRootD storage (FAX) [9] which aims at providing a unified interface to the different ATLAS storage systems. The File Residency Manager is responsible for downloading files located through a request to the global redirector and storing it in the XRootD cluster, and also for purging seldom unused files from it.

2.4. CernVM-FS
CernVM-FS [10] (CVMFS) is a HTTP based file system that provides access to experiment software through caching and downloading on demand. In this way the operating system is effectively separated from the experiment software, and consistent access regardless of the machine type and operating system is provided. CVMFS repositories are provided for ATLAS software, for generic grid software, and for the other experiments on the LHC. The software repositories can be accessed directly from the so-called Stratum 1 servers, but CVMFS recommends the use of a Squid [11] service to cache software close to the client, and it is mandatory for clusters with more than 30 simultaneous connections.

3. Contextualizing machines on GCE
The process of contextualization is that of configuring a machine for its role at the first time it is started. Contextualization as discussed in this paper entails anything from setting up extra disk space attached to virtual machine instances, installing software, configuring software, and starting software services. Once the contextualization step is complete the machine is ready to assume its role. On most commercial and private clouds contextualization can be achieved in two ways; either by creating a custom machine image that is uploaded to the cloud, or through the means of instance or cloud metadata provided at instantiation which might include user data. It is also possible to use commands over e.g. SSH [12] to contextualize a machine, provided that connectivity allows.

On GCE it is not possible to upload custom machine images, meaning that tools such as Boxgrinder [13] or HEP-adapted appliances such as CernVM [14] can not be used. Instead the entire process of creating a custom machine image has to take place on a machine running inside the GCE. This severely hampers versatility by causing even small and predictable configuration changes to require starting a machine and applying configuration steps manually. GCE supports contextualization via metadata, and this then becomes the only viable option. This metadata can take the form of a program that is executed on the machine during boot. Although such a program can theoretically perform any configuration task, it would require a non-negligible development effort, and there already exist tools suitable for the task.
3.1. Puppet configuration
We use Puppet to configure the machines in the cluster. Puppet offers a masterful setup, where one node acts as Puppet master which distributes configurations to other nodes. In this setup configuration changes are pushed to the Puppet master and nodes are automatically updated to reflect the changes. Puppet also offers a masterless setup where configuration changes are pulled to and applied on each node. We chose to use the masterless setup and to distribute the Puppet configuration with Git [15] via a public GitHub [16] archive. Updates are achieved by sending commands over SSH [12] to pull updates from the Git repository and apply them locally. For this method of updating to work, the machines must have a public IP address, an SSH service running, and the corresponding port must be open for inbound connections.

The Puppet configuration is separated into modules where each describes the desired state of a specific function or software. It includes modules for setting up CVMFS, XRootD, HTCondor, and AutoPyFactory, and it also contains modules for installing software packages necessary for ATLAS analysis. Each module is configurable with a set of parameters making it reusable in other contexts. The configuration is available to the general public from the atlasgce-modules [17] GitHub repository.

One of the benefits of using Puppet is that the modules contain all the necessary information to configure a node as either a manager or a worker. It also makes it easy to add further roles to extend the capabilities of the modules. What role a node should assume can thus not be a part of the modules themselves, but must be provided in the form of a node configuration template by the means of instance metadata. (This is also how any secrets must be provided, since they should not be part of the publicly available modules. Modification of instance metadata is secured with HTTPS.) This node configuration template is a top level component that contains items that differ between individual cluster, such as PanDA cloud, site, and queue, XRootD global redirector, and CVMFS proxy configuration.

3.2. Bootstrapping
Just providing a Puppet configuration in the form of modules and node configuration template is not enough to contextualize a machine running a completely unconfigured machine image. We must also provide a way to put Puppet on the machine, and tell it what modules and node configuration template to use. This procedure of preparing a machine for more extensive configuration is called bootstrapping.

Bootstrapping a node on the cluster is done by a program designed to do as little as necessary and in as simple a way as possible. It is executed as part of the operating system startup procedure and performs the following tasks: it installs Puppet and Git; it formats and mounts any extra disk space attached to the machine; it downloads the Puppet modules; it applies the node configuration template which finalizes the contextualization. The bootstrapping procedure is executed at a late stage during boot, when networking and other necessary functions have become available.

4. Results
Our investigations show that it is possible to contextualize machines on GCE with Puppet for ATLAS analysis. It also shows that a single set of Puppet modules can serve configurations for several different roles in a cluster. The modules along with the node configuration templates for each role give a consistent description of the cluster collected in one place, written in one language. The modules are publicly available and fully documented in the atlasgce-modules [17] GitHub repository.
Table 1. Cluster startup time. The times given for the node instantiations are from GCE has registered the command until the GCE reports that the node has started. They do not include contextualization times.

| Event                           | Time | Acc. time |
|---------------------------------|------|-----------|
| Cluster start command           | 4 s  | 4 s       |
| Head node instantiation         | 43 s | 47 s      |
| Worker node instantiation (1-4)| 29 s | 76 s      |

Table 2. Node contextualization times of bare CentOS 6 images. The times are defined as the difference between the start and completion times of the different steps of the bootstrap script that is running the contextualization as they are listed in the operating system log files.

|                      | Head     | Worker 1 | Worker 2 | Worker 3 | Worker 4 |
|----------------------|----------|----------|----------|----------|----------|
| Installing Puppet    | 0:40 min | 0:32 min | 0:32 min | 0:30 min | 0:37 min |
| and support          |          |          |          |          |          |
| Formatting additional storage | 0:42 min | 1:15 min | 1:22 min | 1:21 min | 1:09 min |
| Applying Puppet      | 4:16 min | 4:17 min | 3:54 min | 3:56 min | 4:25 min |
| configuration        |          |          |          |          |          |
| Total                | 5:38 min | 6:04 min | 5:48 min | 5:47 min | 6:11 min |

4.1. Performance of GCE

GCE has provided a stable platform for the setup and testing of the cluster, and the development of the Puppet configuration and the supporting scripts. Although no long-term availability tests have been made during the this study, the cluster has been available for the entirety of the development; a period of four months.

An earlier study [18] used a setup similar to what is described in this paper to run ATLAS production jobs corresponding to about 214 million events during a period of 8 weeks using 4000 cores. During this study GCE proved a very stable platform and none of the job failures that occurred were due to GCE.

4.2. Impact on startup times

Contextualizing a machine from scratch rather than using pre-configured machine images does of course have a detrimental effect on machine startup times. However the additional time required to download, install, and configure software and services is not so large that it affects its usefulness for on-demand resource creation. The contextualization times of a cluster consisting of a head node and four worker nodes are listed in Table 2. The main part is spent downloading software packages in the Applying Puppet configuration step.

4.3. Inbound performance of XRootD

The inbound performance of XRootD from FAX to GCE nodes has also been tested. Average transfer rates for transferring from a single source amounts to 40 MB/s, and from multiple sources using multiple data streams to 57 MB/s. This performance is achieved over public networks and using an XRootD cluster with 1.7 TB disk space per node [18].

No other metrics pertaining to file transfer and caching have been measured. In particular no measurement of the outbound performance of XRootD has been made.
5. Conclusion
We have investigated how well Google’s IaaS platform GCE is able to perform as an ATLAS analysis cluster. GCE launches instances very quickly which suits an on-demand analysis cluster scheme particularly well. A prior study [18] has shown that GCE is also able to handle the large amount of input data that is required for ATLAS analysis. With extra disk space serving as a local XRootD cache GCE also becomes a possible option for a semi-permanent cluster.

We have also investigated the use of Puppet to contextualize computing resources for ATLAS analysis clusters in particular, and more generally for any computing activity in HEP that uses one or more of CVMFS, HTCondor, or XRootD. Puppet has been found to function well in this context: startup times increase but remain manageable, and the benefits are numerous. Puppet offers a clear and comprehensive way to specify a machine configuration that can be made to work with multiple machine roles, on any operating system, and with any cloud provider. By extension it is possible to contextualize machines in any grid site, or other system of machines in the same way. The increased startup times can be reduced by using machine images where the required software has been pre-loaded, but where configuration is still handled by Puppet. Indeed, the very same Puppet configuration can be used as for the bare machine images.

With IaaS clouds it is trivial to add or remove machines from a cluster, and with Puppet it is easy to adjust the configuration without terminating and restarting it. The possibility to update the configuration of running machines – even reconfigure an entire cluster – is a valuable improvement over image based contextualization methods.

References
[1] ATLAS Collaboration 2008 \textit{JINST} \textbf{3} S08003
[2] Puppet Labs: IT Automation Software for System Administrators (Accessed: 2013-10-01) URL \texttt{http://puppetlabs.com/}
[3] Google Compute Engine — Cloud Platform (Accessed: 2013-10-01) URL \texttt{https://cloud.google.com/products/compute-engine}
[4] ATLAS Collaboration 2008 \textit{J.Phys.Conf.Ser.} \textbf{119} 062036
[5] ATLAS Collaboration 2011 \textit{J.Phys.Conf.Ser.} \textbf{331} 062040
[6] Caballero J, Hover J, Love P and Stewart G A 2012 \textit{J.Phys.Conf.Ser.} \textbf{396} 032016
[7] Thain D, Tannenbaum T and Livny M 2005 \textit{Concurrency - Practice and Experience} \textbf{17} 323–356
[8] Dorigo A, Elmer P, Furano F and Hanushevsky A 2005 \textit{WSEAS Transactions on Computers} \textbf{1}
[9] About | Federated ATLAS storage using XRootD (Accessed: 2013-10-09) URL \texttt{http://atlas-xrd.web.cern.ch/content/about}
[10] Blomer J, Buncic P, Charalambidis I, Harutyunyan A, Larsen D and Meusel R 2012 \textit{J.Phys.Conf.Ser.} \textbf{396} 052013
[11] squid : Optimising Web Delivery (Accessed: 2013-10-01) URL \texttt{http://www.squid-cache.org/}
[12] The Secure Shell (SSH) Protocol Architecture (Accessed: 2013-10-01) URL \texttt{http://www.ietf.org/rfc/rfc4251.txt}
[13] BoxGrinder (Accessed: 2013-10-01) URL \texttt{http://boxgrinder.org/}
[14] Buncic P, Aguado Sanchez C, Blomer J, Franco L, Harutyunyan A, Mato P and Yao Y 2011 \textit{J.Phys.Conf.Ser.} \textbf{331} 052004
[15] Git (Accessed: 2013-10-01) URL \texttt{http://git-scm.com/}
[16] GitHub (Accessed: 2013-10-01) URL \texttt{https://github.com}
[17] atlasgce-modules (Accessed: 2013-10-01) URL \texttt{https://github.com/spiiph/atlasgce-modules}
[18] Cloud Computing and High-Energy Particle Physics: How ATLAS Experiment at CERN Uses Google Compute Engine in the Search for New Physics at LHC (Accessed: 2013-10-01) URL \texttt{https://developers.google.com/events/io/sessions/333315382}
[19] By the numbers: How Google Compute Engine stacks up to Amazon EC2 (Accessed: 2013-10-01) URL \texttt{http://gigaom.com/2013/03/15/by-the-numbers-how-google-compute-engine-stacks-up-to-amazon-ec2/}