Journal of Physics: Conference Series **513** (2014) 062040 doi:10.1088/1742-6596/513/6/062040

A Voyage to Arcturus: A model for automated management of a WLCG Tier-2 facility

Gareth Roy[1], David Crooks[1], Lena Mertens[1], Mark Mitchell[1], Stuart Purdie[2], Samuel Cadellin Skipsey[1], David Britton[1]

[1]Department of Physics and Astronomy, University of Glasgow, G12 8QQ
[2]University of St Andrews

E-mail: gareth.roy@glasgow.ac.uk

Abstract. With the current trend towards "On Demand Computing" in big data environments it is crucial that the deployment of services and resources becomes increasingly automated. Deployment based on cloud platforms is available for large scale data centre environments but these solutions can be too complex and heavyweight for smaller, resource constrained WLCG Tier-2 sites. Along with a greater desire for bespoke monitoring and collection of Grid related metrics, a more lightweight and modular approach is desired. In this paper we present a model for a lightweight automated framework which can be use to build WLCG grid sites, based on "off the shelf" software components. As part of the research into an automation framework the use of both IPMI and SNMP for physical device management will be included, as well as the use of SNMP as a monitoring/data sampling layer such that more comprehensive decision making can take place and potentially be automated. This could lead to reduced down times and better performance as services are recognised to be in a non-functional state by autonomous systems.

1. Introduction

There are multiple approaches to the automation of services within the computing environments provided by the WLCG and EGI organisations. At present there is no standard way to build, monitor and operate a Tier-2 computing facility. With the advent of Cloud services, Virtualisation, Software Defined Networking and other deployment technologies the complexity in delivery Grid services to research environments has become so large that individual management of systems is no longer possible. A method to simplify and automate the deployment of Grid systems is essential. The commercial sector has solved this problem by moving much of its provisioning and deployment mechanisms into the cloud, offloading much of the physical hardware support to commercial providers. Using software platforms such as the Apache MESOS [2] they can automate the deployment of large applications consisting of many distributed software components.

A careful study is required of the components necessary to build and operate a WLCG Tier-2. In this paper a model is proposed for an automated framework that can be used as a platform to allow Grid services to be run. It consists of tools for provisioning both the "Bare Metal" and virtualised containers, configuration management and a unified control and monitoring solution. This set of tools can be scripted to allow automated deployment and configuration of an appropriately selected Grid software stack to be spun up at ease. An initial survey into the

use of the tools presented in this paper has been carried out using resources located at ScotGrid Glasgow [1].

2. Provisioning

There are two distinct phases in the construction of a WLCG Tier-2 site. The first is to provision the physical hardware to create a platform for building the complex software components required, the second is to partition the resources and assign specific roles by installing appropriate software components. In the past, specific roles were assigned to individual pieces of hardware; a distinction was drawn between a cluster head node, a pool node or a compute element (CE) for example. Each individual server had its own unique set of build steps and required individual attention. To automate the configuration and management of these services roles need to be abstracted from the underlying physical hardware. This can be accomplished by leveraging available tools for "Cloud" deployments and by virtualising the system on which the software will run.

2.1. Bare Metal Provisioning

The initial phase of provisioning physical hardware is often referred to as "Bare Metal" provisioning. Most Grid sites in the WLCG use a derivative of Redhat Enterprise Linux (RHEL) to provide their software services. The steps involved in provisioning of such systems are outlined in Figure 1. In this workflow, a machine is configure to us the Preboot Execution Environment (PXE) to query a DHCP server on boot, the DHCP server redirects the machine to a PXEBoot server which transfers a Network Bootstrap Program (NBP) via the Trivial File Transfer Protocol (TFTP). Once the initial NBP is downloaded and executed it attempts to carry out installation of the Linux system by requesting a Kickstart file which contains a list of instructions on how to partition disks, configure the network and which packages to install from a remote repository.

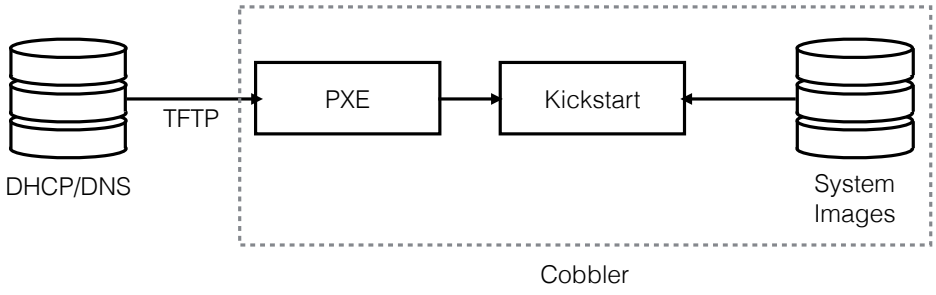

Figure 1. Data flow followed by Linux "Bare Metal" provisioning

A number of components are required to make the workflow depicted in Figure 1 complete; a TFTP server is needed, Kickstart files need to be written and maintained, system images must be created, an initial NBP must be created. A range of software tools are available that manage the diverse components needed to provision physical hardware. For derivatives of RHEL, the most suitable appears to be Cobbler [3]. Cobbler wraps much of the process flow found in Figure 1 and can be allowed to manage the DHCP/DNS resolution as well as managing the TFTP server; it can also be remotely scripted via a XMLRPC API.

2.2. Cloud Provisioning

After "Bare Metal" provisioning the hardware needs to be partitioned and assigned roles. As stated earlier industry has solved the problem of partitioning by relying on technologies such as

Xen, KVM and VMWare to perform system level virtualisation. Here a virtual machine (VM) is created so that multiple tenant operating systems can be run on a single piece of hardware. One downside of this approach is a reduction in performance related to the virtualisation of architectures. Although with CPU extensions this can be ameliorated, it cannot be completely eliminated and in particular performance of code heavily IO dependent can be badly affected.

An alternative, newly popular, technology is Linux Containerisation. This technology is similar in scope to BSD jails and can be described as operating system virtualisation. In this case the Linux Kernel itself can virtualise and isolate a user-land level instance of the operating system in which services can be run. While complete isolation of tenant code is lost due to the shared Kernel, performance is improved as no translation is necessary. Through the use of *cgroups*, which allowed resource limitation and namespace isolation a complete filesystem can be virtualised in a similar fashion to that found in VM's.

There are many competing software packages for working with containers; Google has recently released their own tools [4], with Redhat providing another solution [5] which uses SELinux to carry out better process isolation. A utility named Docker [6] is promising with support for Redhat having being obtained. While cgroups and containers have been part of the Linux Kernel from 2.6.24 many of the tools listed above are new and are not at the point of being production ready. A method of working with containers that is more stable (but less full featured) is included in the standard Scientific Linux distribution via LibVirt [7]. This is an open source library which provides an API and a set of tools for working with many Virtualisation platforms. Since many utilities and tools are already capable of working with LibVirt this is the most promising technology for creating, managing and working with Linux containers in a production environment.

3. Configuration Management

Once hardware is provisioned and partitioned it is necessary to configure the system and install Grid software. Figure 2 shows one possible model for automated configuration management. In this example a configuration management tool obtains a set of template configurations from a versioned configuration store (such as from a Subversion or Git repository). It also obtains service descriptions and information from a Service Discovery service similar to Apache Zookeeper [8] and the CoreOS etcd [9].

A variety of tools exist for remote configuration, such as Puppet [10], Chef [11] or CFEngine [12]. Currently most EMI software is configured via Yaim [13], a utility for generating configuration files built around a standard set of inputs. Puppet is becoming popular amongst grid sites for general system configuration and indeed some Grid Middleware components such as DPM are including standalone configurations in Puppet format to replace Yaim.

While Puppet is a good solution to the distributed configuration problem it follows a "pull" model where individual agents residing on each machine call into a central Puppet Master for their configuration, this requires Puppet, Ruby and other components to be installed locally on all managed nodes. Additionally Puppet uses a PKI to identify nodes adding additional complexity.

An alternative solution to be considered is provided by Ansible [14]. Ansible follows a "push" model with configurations "pushed" to a node. The only requirement is that the end point has Python installed and is accessible via SSH. This is a more lightweight solution than Puppet and can co-exist with other tools, so is a good candidate for system configuration.

4. Monitoring and Control

To create an automation framework it is essential to have the ability to interact and manage physical hardware devices present in the Tier-2. In this section the use of both IPMI and SNMP

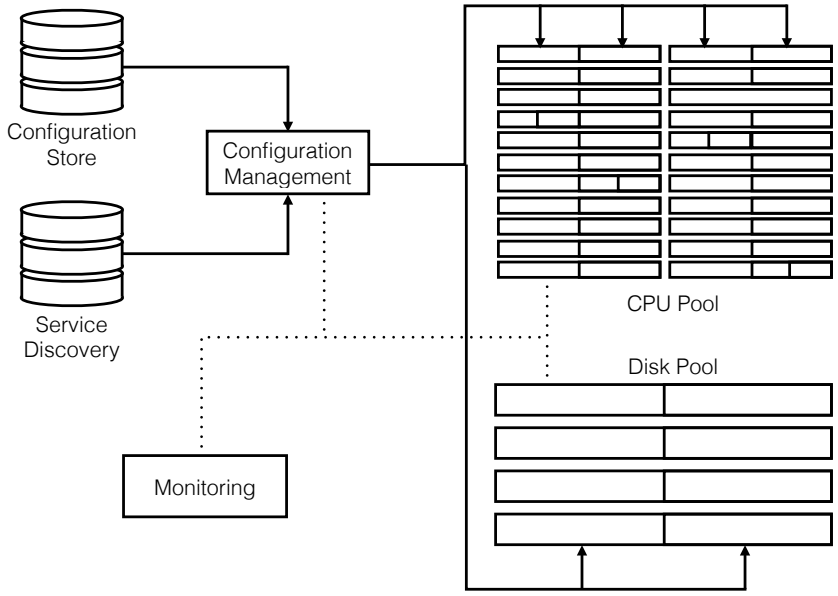

Figure 2. Model of automated configuration management.

for physical device management is explained, along with the use of SNMP as a monitoring/data sampling layer to allow automated decision making.

4.1. Simple Network Management Protocol

Simple Network Management Protocol (SNMP) [15] is a protocol that allows the management of network attached devices. SNMP consists of a management interface that communicates with agents residing on the managed device. These agents can be queried to provide configuration information and carry out stored operations (such as the activation or deactivation of a port on a switch).

SNMP itself does not define the information that can be obtained from an agent but instead use Management Information Bases (MIBs) which contains a hierarchical list of Object Identifiers (OIDs) that represent variables or actions that can be carried out. As with other software mentioned in this paper there are a range of libraries that can be used to interact with SNMP, however the most popular of these is Net-SNMP [16]. In Figure 3 a fragment of code is presented that uses Python bindings provided by Net-SNMP to query a PDU for the current being drawn on each of its banks and the total current drawn.

The Net-SNMP library provides a comprehensive set of tools that can be used to interact with SNMP agents, and includes the ability to run agents on linux servers. It also allows the creation of SNMP traps that allow agents to send messages to the management interface when criteria are met such as high load or quotas reached.

4.2. Intelligent Platform Management Interface

While SNMP is a good approach for managing network attached devices it is unable to manage servers that are unresponsive or in fact have been turned off. A second complimentary system needs to be used to achieve this with the most popular currently being the Intelligent Platform Management Interface (IPMI). IPMI is a standardised system interface (initially created by Intel [17]) which allows out of band management of a hardware platform. It relies on additional hardware being present in the form of the Baseboard Management Controller (BMC) which

```
import netsnmp

session = netsnmp.Session(DestHost='10.141.127.254',
                          Version=1, Community='public')

total = netsnmp.Varbind('PowerNet-MIB::rPDULoadStatusLoad.1')
bank1 = netsnmp.Varbind('PowerNet-MIB::rPDULoadStatusLoad.2')
bank2 = netsnmp.Varbind('PowerNet-MIB::rPDULoadStatusLoad.3')

result  = session.get(netsnmp.VarList(total,bank1,bank2))
```

Figure 3. Example code fragment using the Python programming language and NetSNMP library to access the current drawn from a metered PDU using SNMPv1.

allows remote access to such things as the BIOS and power management settings. A system can be powered and the boot process monitored via access to the BMC.

5. Conclusion

In this paper a set of tools has been described that can be coupled to create an automated framework for the deployment of Grid services. Tools for provisioning "Bare Metal" and virtualised containers, configuration management and a unified control and monitoring solution have been described. Each component has been investigated on a test cluster located at ScotGrid Glasgow. The tools described here will be used in future work to develop a set of scripts that will allow the automated deployment of software services and allow the creation of an API would allow for an Apache MESOS style deployment of Grid Middleware.

References

[1] *ScotGrid* [online] Available at http://www.scotgrid.ac.uk/ [accessed 16 January 2014]
[2] *Apache MESOS* [online] Available http://mesos.apache.org/ [accessed 16 January 2014]
[3] *Cobbler* [online] Available http://www.cobblerd.org/ [accessed 16 January 2014]
[4] *Let me contain that for you* [online] Available https://github.com/google/lmctfy [accessed 16 January 2014]
[5] *Redhat OpenShift Gears* [online] Available https://www.openshift.com/ [accessed 16 January 2014]
[6] *Docker.io* [online] Available https://www.docker.io [accessed 16 January 2014]
[7] *LibVirt: The Virtualisation API* [online] Available http://libvirt.org/ [accessed 16 January 2014]
[8] *Apache Zookeeper* [online] Available http://zookeeper.apache.org/ [accessed 16 January 2014]
[9] *The etcd Project* [online] Available https://github.com/coreos/etcd [accessed 16 January 2014]
[10] *Puppet* [online] Available http://puppetlabs.com/ [accessed 16 January 2014]
[11] *Chef* [online] Available http://www.opscode.com/chef/ [accessed 16 January 2014]
[12] *CFEngine* [online] Available http://cfengine.com/ [accessed 16 January 2014]
[13] *Yaim Aint' an Installation Manager* [online] Available https://twiki.cern.ch/twiki/bin/view/EGEE/YAIM [accessed 16 January 2014]
[14] *Ansible* [online] Available http://www.ansibleworks.com/ [accessed 16 January 2014]
[15] *Internet Engineering Task Force (IETF) RFC 3411* [online] Available http://tools.ietf.org/html/rfc3411 [accessed 16 January 2014]
[16] *Net-SNMP* [online] Available http://www.net-snmp.org/ [accessed 16 January 2014]
[17] *IPMI* [online] Available http://www.intel.com/content/www/us/en/servers/ipmi/ipmi-home.html [accessed 16 January 2014]