Upgrade and integration of the configuration and monitoring tools for the ATLAS Online farm

S Ballestrero[1,2], F Brasolin[3], G–L Darlea[1,4], I Dumitru[4], D A Scannicchio[5], M S Twomey[1,6], M L Valsan[4] and A Zaytsev[7]

[1] CERN CH-1211 Genève 23, Switzerland
[2] University of Johannesburg Department of Physics, PO Box 524 Auckland Park 2006, South Africa
[3] INFN Sezione di Bologna, Viale Berti Pichat 6/2, 40127 Bologna, Italy
[4] Politehnica University of Bucharest, Splaiul Independenţei 313, cod 060042, sector 6, Bucharest, Romania
[5] University of California, Irvine, CA 92697, USA
[6] University of Washington Department of Physics, Box 351560 Seattle WA 98195-1560, USA
[7] Russian Acad. Sci., Siberian Div., Budker Inst. of Nuclear Physics (BINP), 11 Academician Lavrentiev prospect, Novosibirsk, 630090, Russia

E-mail: `atlas-tdaq-sysadmins@cern.ch`

Abstract. The ATLAS Online farm is a non-homogeneous cluster of nearly 3000 systems which run the data acquisition, trigger and control of the ATLAS detector. The systems are configured and monitored by a combination of open-source tools, such as Quattor and Nagios, and tools developed in-house, such as ConfDB.

We report on the ongoing introduction of new provisioning and configuration tools, Puppet and ConfDB v2, which are more flexible and allow automation for previously uncovered needs, and on the upgrade and integration of the monitoring and alerting tools, including the interfacing of these with the TDAQ Shifter Assistant software and their integration with configuration tools.

We discuss the selection of the tools and the assessment of their functionality and performance, and how they enabled the introduction of virtualization for selected services.

1. Introduction

The ATLAS[1] Online farm is a complex computing farm, with very high availability requirements. It is composed of nearly 3000 PCs; the majority is part of the High Level Trigger (∼1600 PCs) and of the readout and data flow systems[2]; but a considerable fraction is dedicated to specific applications. Almost all the systems in the farm run Scientific Linux CERN 5, a derivative of RedHat Enterprise Linux 5. This number of systems and their variety require appropriate tools to be managed effectively with our limited manpower. Where Open Source tools existed, they have been adopted and extended if needed; for more particular requirements we have developed our own tools.

The majority of the Linux systems boot the Operating System via network. This allows for a uniform, single image base system, to which specific configurations can be applied if the operational role of the node requires them. We call these NetBooted Nodes; they are managed by our own ConfDB[3] system and the Boot With Me tool chain.

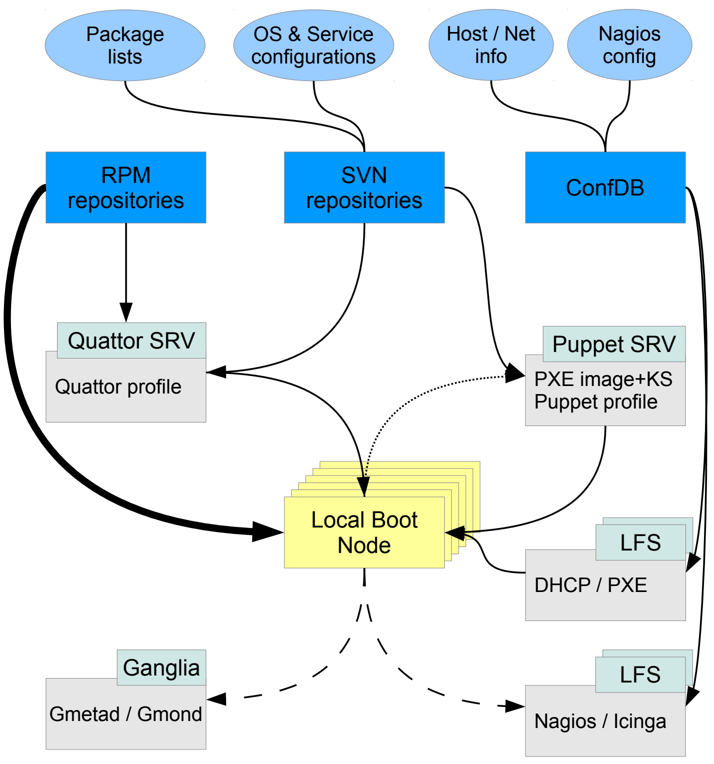

Figure 1. A node with locally installed OS receives configuration information from various sources (thin arrows); the Puppet server uses node status information (dotted arrow) to compile the profile. Its OS installation and updates are provided directly by the repository (thick arrow). Monitoring information is collected by one of the Nagios servers and more detailed performance information is sent to the dedicated Ganglia server (dashed arrows).

A certain number of systems have requirements that cannot be easily met by, or contrast with, booting the OS via network. For these the OS is installed on a local disk; some aspects are managed by ConfDB, but mostly the configuration is managed by Configuration Management Tools (CMT) - Quattor and Puppet.

The limited number of systems that do not run Linux are managed by other teams and will not be discussed in this paper.

2. LocalBooted nodes

Most special purpose nodes in the ATLAS Online Farm have the operating system installed on a local disk. With respect to the NetBooted system that will be discussed in a later section, this type of configuration is more vulnerable to disk failure, but is more resilient to transient network failures and allows considerably more flexibility. We use it primarily for various types of servers (DHCP, DNS, MySQL, LDAP, webservers, Application Gateways, Central File Server, Local File Servers and various others), for special purpose clients (Detector Control System, ATLAS Control Room, Secondary Control Room, Public use nodes) and for the nodes hosting Virtual Machines, for a total of ∼260 nodes.

The provisioning of local boot nodes, either physical host or virtual, is done via a PXE boot of SLC in install mode. The DHCP and some PXE boot information are provided by a Local File Server, from configurations generated by ConfDB.

The PXE options, the boot image and the “Kickstart” configuration file for the Anaconda installer are provided by one of the CMT servers. The host-specific Kickstart is generated by a simple script system that uses `m4` templates. The initial setup is limited to the partitioning and formatting of the OS disk, and the installation of a minimum system to start the configuration management system chosen for the given host.

Afterwards, the CMT takes control of the system and installs the packages required for the applications, and configures network, authentication, services etc. Figure 1 is a diagram of the overall flow of information for LocalBooted nodes.

2.1. Quattor

Quattor[4] has been the Central Configuration Management Tool of choice for CERN. It has good functionality, not too distant from CFEngine[5], and offers a very good guarantee of consistency of the system profile generated from its template descriptions which use the PAN declarative language; it is well integrated with Software Package Management Agent (SPMA), a tool that provides detailed and tight control over the RPM software packages installed on a system. It also has relatively good scalability and low impact on the client, since the profiles are compiled just once on the server and then provided to the clients as static files via HTTP.

In ATLAS we have used Quattor extensively, and, as of 2012, about 240 of the local install nodes in the Online farm have been installed and are managed with Quattor.

The difficulties and limitations of Quattor started to emerge, in 2010, while attempting to do fully automated installations, down to minor details, and became more evident when we attempted to do central management of systems with more complex configurations, such as the ATLAS Operations web servers. The main issue has been the inflexibility of the execution sequence of modules, which makes it impossible to combine basic operations into complex sequences (e.g. install AutoFS configuration files, start automount, rsync a directory tree from an NFS server, start another service) and requires instead the creation of modules ex-novo; writing modules requires an advanced knowledge of Quattor and the functional language for the PAN compiler, and a very careful implementation of the client-side part of the module, usually in a different, imperative and sequential language such as Perl. The other issues that prompted us to look for alternatives to Quattor are the time consuming management of system updates and the inability of SPMA to automatically handle package installation dependencies.

2.2. Puppet

We investigated various possible replacements for Quattor; besides overcoming the described difficulties encountered with Quattor, we had a strong interest in finding a tool that would offer a gentler learning curve, so that it could be used not just by one or two experts but by all members of the team, including short term collaborators, and possibly dispel the temptation of doing manual configurations. As usual we prefer Open Source software, which we can independently adapt to our requirements if necessary; after a first selection of the projects that were actively developed and supported, the most promising ones were CFEngine, Chef and Puppet.

CFEngine[5] has been for a long time the reference CCMT in the IT industry, and is widely adopted. It has a clear philosophy, that system configurations are described by "guarantees" expressed in a declarative language[6]; has a clear design, is known for having a good scalability, performance and low impact on the client. Its weak points, for our expected usage, are the verbose definitions and the limited internal functionality that often requires writing too much code, even for simple operations.

Puppet[7] is in more than one way inspired by CFEngine, and places a strong accent on the concept of idempotence; in the system configuration context this means that applying, even repeatedly, a given configuration produces a converging, consistent result. It has a good arsenal of predefined "types", which can be easily combined to describe complex configurations, and an easy templating mechanism based on Ruby's ERB; it also integrates very well with the powerful

Augeas[8], an abstract, generalized tool for automated editing of configuration files. The Domain Specific Language of Puppet is purely declarative, with no intermixing of functional or procedural; while it may feel unfamiliar at first, it also gives strong indications when one tries to force Puppet against its intended usage. The weak points of Puppet are a relatively high resource usage on the client, and its limited scalability when used in the complete client-server mode. In the basic server-less mode Puppet is instead trivially scalable.

Chef[9] is the newest of the three, and also in many ways inspired by its predecessors. It does offer some more flexibility that Puppet, but it uses a more verbose and complex Ruby-based, imperative language, and has a more limited functionality “out of the box”; it generally appears more oriented towards a public of developers willing to automate application deployment, than to system administrators. Its scalability to large numbers is also less immediate than the obvious scalability of server-less Puppet.

From the considerations above, and other comparisons, e.g. [10], we chose Puppet and started to use it for the management of the kind of complex, often “one-off” special purpose servers that were previously managed manually. Since 2010 we gained confidence and came to trust Puppet to behave correctly and reliably, we have increased the number of Puppet modules for service and OS configurations, and the majority of the members of the team is familiar with Puppet. During 2011 we have started to apply Puppet also to production systems that are still primarily managed by Quattor, allowing a smooth transition path with no impact on the production farm. The migration will continue during 2012, with the goal of using Puppet exclusively for SLC6, which will go in production in 2013, during the LHC Long Shutdown.

3. NetBooted nodes

The vast majority of the nodes in the ATLAS Online Farm, almost 2400, are booted into Scientific Linux via PXE. This method of OS booting is necessary for diskless systems such as VME-mounted Single Board Computers (SBC), but it also has advantages for other systems. For certain critical systems which do not need local storage, like the PCs of the ReadOut System (ROS) sub-farm, the absence of a hard disk removes a possible point of failure. For large scale sub-farms, such as the HLT, a local disk can be used as scratch space, and its failure can be recovered quickly without a reinstall. But, possibly more important, the usage of a single image provides a strong guarantee of uniformity and consistency across large numbers of systems, and the easy option of simply rebooting to start again from a pristine system, without a time-consuming OS reinstall.

As shown in Figure 2, the so called Local File Servers (LFS) provide DHCP and TFTP for the boot process, and the `/usr` hierarchy which is mounted via read-only NFS by the clients. There are $\sim$80 LFS, each serving typically $\sim$30 clients. On the LFS, the DHCP and PXE configurations are extracted from the ConfDB system[3].

3.1. Boot With Me

Boot With Me is a specialized software tool used in ATLAS Online to create the necessary elements for the PXE booting of a client host. BWM uses a complete system image (technically the disk image of a non-running Virtual Machine) as the reference “golden image”, to generate a bootable PXE image and a `/usr` filesystem hierarchy. The bootable image includes the kernel, a boot-time disk image, and an image of the root filesystem that is mounted as a RAMdisk and used at run time. The kernel version, boot disk and content of the root filesystem are controlled by configuration files, so that it is possible to create different PXE boot images from the single golden image, for example a standard image, suitable for production use on all systems in ATLAS, or a full-featured image, suitable for tests, firmware updates etc.

We use two separate Virtual Machines as BWM “build” systems; one for the common 64-bit SLC5, and one for the 32-bit SLC5 which is needed for certain systems in the Online farm -

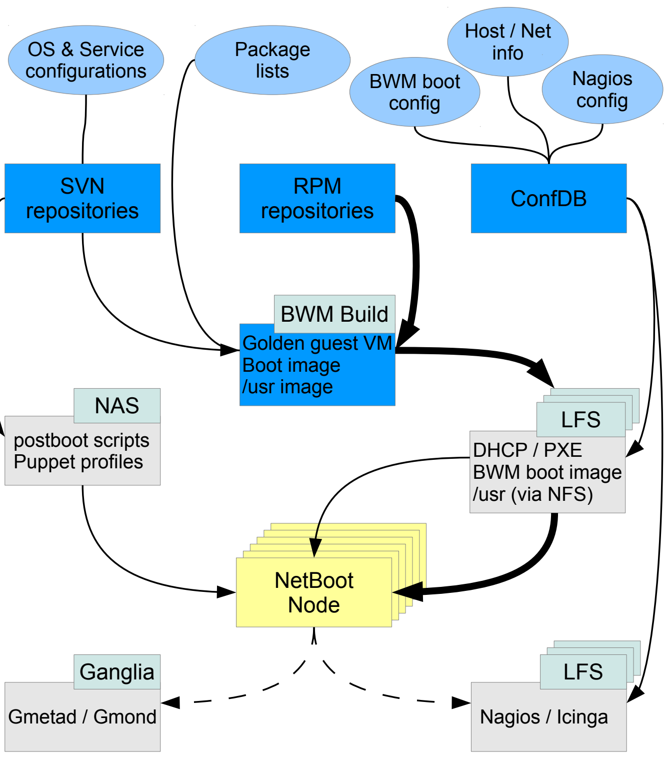

Figure 2. A NetBooted node receives OS and configuration information from various sources (thin arrows). Its boot and operating system are provided by BWM via its LFS (thick arrows). Monitoring information is collected by a Nagios server (usually the boot LFS); some hosts also send more detailed performance information to the dedicated Ganglia server (dashed arrows).

primarily for the SBC and the ReadOut System (ROS) farm PCs. The generated boot images and `/usr` are copied to the ATLAS Online software distribution host, the Central File Server, and from there propagated via rsync to all the LFS.

Both the BWM software and its configurations are maintained in a Subversion repository; the contents of the reference system image are not yet subject to a version control system, but for any major release we archive a full image snapshot, to allow a fast rollback of changes. The reference system itself is a pristine, unmodified installation of Scientific Linux CERN that can be rebuilt easily by performing a default installation on a Virtual Machine.

3.2. BWM Post-Boot

After the OS boot, each node is configured by scripts, mostly written in bash, with host-specific configurations, such as services, disk and NFS mounts etc, according to a hierarchy defined by the hostname. This allows us to customize the standard "production" boot images for the specific application.

The core steering script parses the standardized hostname (e.g. `pc-tdq-onl-53`) into components, considering it in the same fashion as a filesystem path, and applies the specific scripts for each level of the path - a `pc.sh` script, a `pc-tdq.sh` script, `pc-tdq-onl.sh`, and finally, if it exists, the host-specific `pc-tdq-onl-53.sh`. The scripts also provide extensive logging, useful for the debugging of possible issues.

The scripts are executed by the NetBooted client, which reads them from a central NAS (a NetApp 3140) via NFS. Given the small file sizes of the scripts involved, this method is sufficiently scalable, even in occasion of nearly simultaneous boots of most of the farm - for example in case of fast recovery from power cuts.

This BWM post-boot script system is extremely flexible and has worked well in production since before the start of the first LHC run in 2008. The main downside of this system is the lack of guaranteed consistency in the resulting configuration - scripts at different levels of the hierarchy can intervene differently on a certain configuration, and it is not always straightforward to determine the final result by a simple examination of the scripts. Also, the scripts often assume to be running on a freshly booted system, so the only way to ensure that an updated script produces the correct configuration is a reboot of the system, which may be impractical or impossible outside of the rare technical intervention periods. The resulting system is also time consuming to maintain, also because of legacy code that had accumulated in the scripts.

Because of the consideration above, and following the positive experience with Puppet on local boot nodes, we have recently started to migrate some of these post-boot configuration scripts to Puppet profiles. To ensure scalability, we use Puppet in server-less mode, and the profiles are read by the client from the central NAS. A Subversion repository is used to track changes of the BWM image creation configuration, of the post-boot scripts and Puppet profiles.

4. The monitoring system

With the large number of systems in the ATLAS Online Farm, it is a necessity to have the appropriate tools to monitor the health status and performance. The monitoring system currently in use was developed in 2007, based on Nagios[11] v2.5. The existing system primarily monitors the health status of the OS, the hardware and selected services and network components, providing alerting for critical events. A significant development effort went into the detailed monitoring of hardware status via IPMI.

In order to scale up to the expected size of the farm, the system uses a separate Nagios server instance on each of the ∼80 LFS nodes, feeding data to a central storage area for historical data in RRD format, and to a single MySQL Cluster instance (hosted on four servers) for transient work and status data. A web-based interface was developed in-house to display summary and detail data from these central storages.

A task of this magnitude, where we need to manage the configuration of such a large number of independent servers, while avoiding duplications and guaranteeing coverage for all hosts and all services, is something that cannot be handled manually. For this reason the generation of Nagios configurations has been integrated in ConfDB[3], resulting in a system that lets us very efficiently manage the monitoring, with limited loss of flexibility with respect to manual configurations.

This system has proven its effectiveness in production since the start-up of the LHC in 2008, and scaled up with the gradual expansion of the TDAQ farm. In the meanwhile, Nagios and derivatives like Icinga[12] have evolved, and systems like mod_gearman[13] have appeared, which will allow the creation of a scalable Nagios-based monitoring infrastructure based on a single central server and single configuration, avoiding the fragmentation of the current system and the necessity of maintaining our special Web-based UIs[14].

4.1. Ganglia

During 2011 we started to introduce another monitoring system, Ganglia[15], which is more oriented towards performance monitoring and trending. As of May 2012, it is used primarily for special purpose nodes, such as local boot nodes and the Monitoring and Online sub-farms. The architecture of Ganglia, based on a push model, allowed a single central server to easily scale to monitoring 300 hosts. This only required us to replace the direct RRD storage mechanism (which was being limited by the storage IOPS) with the RRD Cache Daemon (rrdcached).

An initial attempt to use the Python implementation of gmetad (which has interesting features like a plugin architecture) was abandoned as it showed a much higher CPU usage, which would have severely limited the scalability.

The data gathered by the Ganglia gmetad collector can also be used as an alternative or additional input to Nagios. This combination would offer both the detailed and scalable data gathering functionality of Ganglia and the advanced alerting capabilities of Nagios. We are currently evaluating this possibility on a smaller scale system; it is very promising in terms of functionality, but the Nagios plugin that we initially chose may not scale nicely, requiring us to consider different implementations.

4.2. Integration with DAQ Shifter Assistant

The DAQ Shifter Assistant[16], developed by the TDAQ Controls and Configuration team, is a tool that supports the person on shift in the ATLAS Control Room, by providing her/him with carefully selected notifications for events requiring human attention, suggesting standard, proven solutions for known problems and alerting if the intervention of an expert is required. Amongst the event streams used by its Complex Event Processing engine is the Nagios monitoring data. Currently, the data for it is extracted by periodically querying the MySQL database; with the planned upgrade to Icinga it will be possible to use ActiveMQ[17] to interface the two systems.

5. Virtualization

The typical workload of the ATLAS Online Farm is generally not suitable for virtualization, since all the CPU power, network and I/O bandwidth must be available for the specialized application running on the specific node. Our team has instead been introducing the use of virtualization for various supporting systems, which have different workload patterns and requirements. Some typical examples are core services, such as Quattor, Puppet and ConfDB, which require the high reliability features of server-class hardware, but would only make use of a fraction of its computing power; running them as isolated VMs, instead of a multiplicity of services in a single OS instance, allows an easier configuration and management, and better control on the resource needs of each service.

Another motivation for virtualization is the increased redundancy and reliability that we can obtain by dividing the workload of a single physical server to two or more VMs; this has been for example the case for the Nagios monitoring dedicated to core, critical hosts. This is supported by the availability of NFS storage space dedicated to the backup of VM images; in case of failure of one physical host, its VMs can be quickly restarted on other hosts. This could possibly be improved to a nearly completely transparent failover with the availability of a fast shared storage system, but the present scale and usage of the small VM farm (6 hosts, 29 VMs) is not sufficient to justify the cost of a suitably reliable storage solution. Using a lower cost storage system without full redundancy would instead introduce an unacceptable single point of failure.

Finally, a strong driving force for virtualization has been the need, particularly for our team but also for other users, to have a considerable number of complete systems for test or development, available for extended periods, or permanently; by using VMs this can be done easily and almost at no cost. This flexibility has considerably improved the work of our team and its capability of developing new systems and solutions.

The new tools, and particularly Puppet, have been extremely important in this virtualization effort, allowing us to manage the provisioning of an increasing number and variety of servers and test systems with limited effort.

Since autumn 2011 following our positive experience with their usage in production, VMs have finally been adopted also for services run by other teams, in particular for the archive of the TDAQ Configuration database (OKSCVS)[18], for the monitoring of the Lucid subdetector, for various monitoring systems of the ATLAS Operations group, and finally as “public terminals” for remote access.

A notable usage of virtualization for ATLAS Online are the Application Gateways, which provide controlled access[19, 20] from the CERN General Purpose Network (GPN) to the isolated ATLAS Technical Network (ATCN) of the Online Farm. Each of the two physical hosts, running the SLC5 Xen Hypervisor, hosts 3 Virtual Machines, providing high redundancy and isolation.

With the exception of the gateways, we are standardizing on the integrated Kernel-based Virtual Machine[21] (KVM), and we are gradually migrating our hosts to SLC6, to take advantage of its improved PXE support and live migration capabilities.

6. Conclusions

The new strategies that we are introducing in the ATLAS Online Farm for configuration management and monitoring are performing very well and are providing the expected improvements in maintainability, flexibility and extending the coverage. In addition, they give better control of the system, and deeper insight into its status. All text/code based configurations are now covered by Subversion, allowing for safe change management, and together with the multiple data sources integrated in ConfDB, this provides a clear structure and a well identified information flow.

The introduction of these new technologies in the production system has been gradual and fully transparent; it will continue during 2012, and will be completed during the LHC Long Shutdown.

References

[1] ATLAS Collaboration, “The ATLAS experiment at the CERN Large Hadron Collider”, J.Instrum.3 S08003 (2008).
[2] ATLAS Collaboration, “ATLAS High-Level Trigger, Data Acquisition and Controls: Technical Design Report”, CERN/LHCC/2003-022, Geneva, CERN, 2003.
[3] ATLAS TDAQ SysAdmins, “Centralized configuration system for a large scale farm of network booted computers”, to appear in Proceedings of CHEP 2012
[4] Quattor: http://quattor.sourceforge.net/
[5] CFEngine: http://cfengine.com/
[6] M. Burgess, “Cfengine: a site configuration engine”, USENIX Computing systems, Vol8, No. 3 1995
[7] Puppet: http://puppetlabs.com/puppet/what-is-puppet/
[8] Augeas: http://augeas.net
[9] Chef: http://www.opscode.com/chef/
[10] T. Delaet, W. Joosen, B. Vanbrabant, “A survey of system configuration tools”, USENIX LISA’10 Proceedings, San Jose, California, 2010
[11] Nagios: http://www.nagios.org
[12] Icinga: http://www.icinga.org
[13] mod-gearman: http://labs.consol.de/lang/en/nagios/mod-gearman/
[14] ATLAS TDAQ SysAdmins, “Tools and strategies to monitor the ATLAS online computing farm”, to appear in Proceedings of CHEP 2012
[15] Ganglia: http://ganglia.sourceforge.net
[16] A. Kazarov, G. Lehmann Miotto, L. Magnoni, “The AAL project: automated monitoring and intelligent analysis for the ATLAS data taking infrastructure”, Proceedings of 14th International Workshop On Advanced Computing And Analysis Techniques In Physics Research, Uxbridge, West London, United Kingdom, 5–9 Sep 2011
[17] ActiveMQ: http://activemq.apache.org/
[18] R. Jones, L. Mapelli, Y. Ryabov, I. Soloviev, “The OKS persistent in-memory object manager”, IEEE Transactions on Nuclear Science 45 (4) (1998) 1958–1964.
[19] ATLAS TDAQ SysAdmins, “System Administration of ATLAS TDAQ Computing Environment”, CHEP 2009
[20] ATLAS TDAQ SysAdmins, “Role Based Access Control system in the ATLAS experiment”, CHEP 2010
[21] Kernel-based Virtual Machine: http://www.linux-kvm.org