ATLAS TDAQ System Administration: an overview and evolution

LEE, Christopher Jon[*†]
University of Johannesburg / CERN
E-mail: chlee@cern.ch

BALLESTRERO, Sergio
University of Johannesburg
E-mail: sergio.ballestrero@cern.ch

BOGDANCHIKOV, Alexander
BINP - Novosibirsk
E-mail: alexander.bogdanchikov@cern.ch

BRASOLIN, Franco
INFN Sezione di Bologna
E-mail: franco.brasolin@cern.ch

CONTESCU, Alexandru-Cristian
University POLITEHNICA of Bucharest / CERN
E-mail: cristian.contescu@cern.ch

DARLEA, Georgiana-Lavinia
University POLITEHNICA of Bucharest
E-mail: gdarlea@cern.ch

KOROL, Aleksandr
BINP - Novosibirsk
E-mail: Aleksandr.Korol@cern.ch

SCANNICCHIO, Diana Alessandra
University of California, Irvine
E-mail: diana.scannicchio@cern.ch

TWOMEY, Matthew
University of Washington / CERN
E-mail: matthew.twomey@cern.ch

VALSAN, Mihai Liviu
University POLITEHNICA of Bucharest / CERN
E-mail: liviu.valsan@cern.ch

The ATLAS Trigger and Data Acquisition (TDAQ) system is responsible for the online processing of live data streaming from the ATLAS experiment at the Large Hadron Collider (LHC) at CERN. The system processes the direct data readout from $\sim$ 100 million channels on the detector through multiple trigger levels, selecting interesting events for analysis with a factor of 10^7 reduction on the data rate with a latency of less than a few seconds. Most of the functionality is implemented on $\sim$ 3000 servers composing the online farm. Due to the critical functionality of the system a sophisticated computing environment is maintained, covering the online farm and ATLAS control rooms, as well as a number of development and testing labs. The specificity of the system required the development of dedicated applications for system configuration and maintenance; in parallel other Open Source tools (Puppet and Quattor) are used to centrally configure the operating systems.
The health monitoring of the TDAQ system hardware and OS performs $\sim$ 60 thousand checks every 5 minutes; it is currently implemented over Nagios, and it is being complemented and replaced by Ganglia and Icinga.
The online system adopted a sophisticated user management, based on the Active Directory infrastructure and integrated with Access Manager, a dedicated Role Based Access Control (RBAC) tool. The RBAC and its underlying LDAP database control user rights from the external access to the farm down to specific user actions. A web-based user interface allows delegated administrators to manage specific role assignments.
The current activities of the SysAdmin group include the daily monitoring, troubleshooting and maintenance of the online system, storage and farm upgrades, and readying systems for an upgrade to Scientific Linux 6 with the related global integration, configuration, optimisation and hardware updates necessary.
In addition, during the 2013 shutdown the team will provide support for the usage of a large fraction of the online farm for GEANT4 simulations of ATLAS.

The International Symposium on Grids and Clouds (ISGC) 2013,
March 17-22, 2013
Academia Sinica, Taipei, Taiwan

[*]Speaker.
[†]On behalf of the TDAQ SysAdmins.

1. Introduction

The Large Hadron Collider (LHC) is the world's largest and highest-energy particle accelerator. It was built by the European Organisation for Nuclear Research (CERN) between 1998 and 2008. On November 23rd 2009 the first recorded proton - proton collisions occurred, at an injection energy of 450 GeV per beam. On the 16th of February 2013, the LHC turned off the accelerator, after achieving 4 TeV per beam. It will remain off for a two year shutdown period (LS1), during which time upgrades will be made to the LHC and its experimental chambers.

During 2012, A Toroidal LHC Apparatus (ATLAS) [1], which is one of the 7 experiments at the LHC, collected 6 PB of raw data, organised into 15M files.This information from ATLAS was captured, processed and stored by the Trigger and Data Acquisition (TDAQ) System [2], through the use of a large online computing farm for the readout of the detector front end data, the trigger decision farms (second and third level of trigger) as well as all the ancillary functions to ensure everything works (monitoring, control, etc.). While in operation, advances in technology allowed the TDAQ system to upgrade many of the design specifications. This resulted in TDAQ recording more than three times the amount of data than was originally intended. This paper takes a look at those changes and also at what is to come.

2. Infrastructure

The physical equipment managed by the TDAQ SysAdmin team [3] is split over two main locations, Point 1 and the CERN Meyrin Site. In first approximation these also correspond to two separate networks, the ATLAS Technical Network (ATCN) and CERN's main General Purpose Network (GPN).

2.1 Point 1 and ATCN

Point 1 is the CERN denomination for the ATLAS Experiment site, where there are two separate equipment rooms.

The first one is located $\sim$ 100 m underground, in close proximity to the detector, in an area called Underground Service Area 15 (USA15). There are 220 equipment racks deployed over 2 floors. In 2009 this was 70% occupied, and consumed 1 MW of power. As of the writing of this paper, this is more than 90% filled and uses about 1.29 MW of power (estimated at $\sim$ 6.5 kW/rack). USA15 hosts the front-end electronics, which occupy the majority of the racks, and the computing equipment that needs to be at a short distance, such as the Detector Control System (DCS), Read-Out System (ROS), and various special-purpose systems.

On the surface, the SDX1 building hosts the main data centre. There are 120 racks, also over 2 floors, which have increased from 50% occupancy in 2009, to about 90% in 2013; 709 kW of power is consumed, compared to 385 kW previously. SDX1 hosts most of the TDAQ farm and core services. In addition, other buildings host the ATLAS Control Room (ACR) and the Satellite Control Room (SCR).

The majority of the systems in Point 1 are connected to the ATCN, which is a private IP network with limited, controlled connectivity to the rest of the CERN network.

2.2 CERN Meyrin Site and GPN

On the Meyrin Site, we have recently commissioned a laboratory which is used by many TDAQ groups to develop and test equipment, networks, system configurations and software prior to their implementation at Point 1. This is called the "TDAQ TestBed". The TestBed infrastructure also provides services to many systems, used by ATLAS sub-detector groups, that are not physically located in the laboratory but can be anywhere on the GPN, both in the Meyrin site and in Point 1.

3. Architecture

The efficiency of the data taking is considered a primary goal; from the point of view of system administration this essentially means avoiding, where possible, any single point of failure in the system and have a fast recovery to minimise the downtime. This approach is what we adopt for critical services; these are typically hosted on server-class hardware with redundant disks, redundant power supplies and independent power feeds (regular and UPS), they have a locally installed operating system and care is taken to minimise their dependency on other services.

For the same reason, but from an opposite perspective, for non critical systems it is important to optimise the ease of maintenance, so that with a limited workload our team can keep the largest possible number of systems in working condition. This approach led us to the netboot method, used for a large number of machines, where TDAQ can also move/remap services (at the application level) in case of failures.

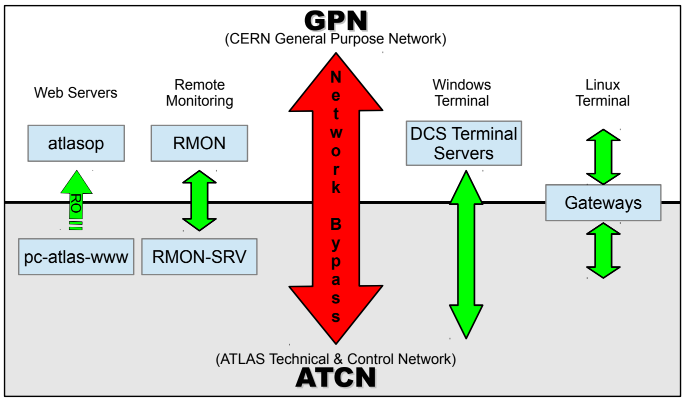

Figure 1: Functional layout of the network separation.

3.1 Network isolation

In addition to the data acquisition systems, the ATLAS Technical Network also hosts the Detector Control System, which includes special purpose network-connected components like Programmable Logic Controllers (PLC), High Voltage Power Supplies etc. These systems are particularly critical for the safety of the detector. It is also for this reason that the ATCN is a network which is mostly separated from the GPN, providing a further level of isolation behind the CERN perimeter firewall.

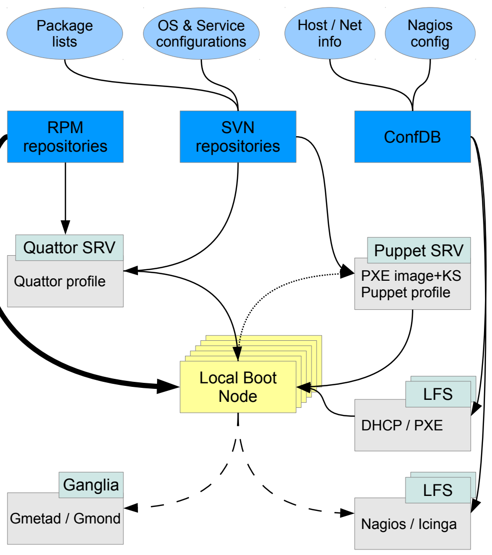

Figure 2: A node with locally installed OS receives configuration information from various sources (thin arrows); the Puppet server uses node status information (dotted arrow) to compile the profile. Its OS installation and updates are provided directly by the repository (thick arrow). Monitoring information is collected by one of the Nagios servers and more detailed performance information is sent to the dedicated Ganglia server (dashed arrows).

As shown in Figure 1, the connectivity between the ATCN and GPN domains is strictly controlled, and only few GPN systems are allowed through the direct network path. User access to ATCN is allowed only through the Gateway systems and the DCS Terminal Servers, all subject to the restrictions of the Access Manager (see Section 6.1). Information that is needed for easy, remote monitoring of the detector status is published read-only either via the Atlas Operations webserver, or via the Remote Monitoring nodes.

One of the requirements in setting up the computing system in Point 1 was to maintain the ability of data taking, in case of a forced disconnection from GPN, e.g. for a major security threat. Therefore all vital IT services (DNS, NTP, DHCP, LDAP, DC) have been duplicated inside ATCN.

3.2 Flat structure with local installed nodes

For critical services, we use systems that have the Scientific Linux CERN OS on the local disk, “localboot” for short. Typically these are server-class systems, with RAID storage. The local installation allows complete flexibility in the configuration, allowing any type of service to be deployed.

Wherever possible, we use redundant systems (DNS, DHCP) or clusters (LDAP) to ensure the service availability in case of hardware failure. In all cases, we ensure a fast recovery by using a provisioning system based on PXE and KickStart for the base OS installation, and Puppet for the full configuration (see Section 4.2). A flow chart of the information flow for this structure can be seen in Figure 2.

The local installed nodes can be broken down as follows:

- Core services and infrastructure: 2 Gateways, 2 DNS+DHCP+NTP, 4 LDAP, 5 MySQL, 3 Central File Servers (CFS), 85 Local File Servers (LFS)
- Control Rooms: 31 ACR and 45 SCR
- Detector Control System: more than 100
- System and network monitoring, virtual hosts etc

A very special role in the system is that of the Central Filer, which supplies the Network File Systems (NFS) and Common Internet File System (CIFS) exports to most of the farm. These include the user home directories, the DAQ Software, Nagios RRD files, node and DAQ configuration areas, static and dynamic web content, as well as a dedicated file exchange area for the Gateways. For this system we have adopted a Network-attached Storage (NAS) system produced by NetApp, the model 3140 which has a dual head controller, and 6 shelves of high-speed disks connected via redundant Fibre Channel paths.

3.2.1 Control Rooms

A noteworthy case in our architecture is the ATLAS Control Room, which is visibly mission critical; for this reason it was decided to adopt server-class computers, and to host them in the controlled environment of the SDX1 room, avoiding at the same time the noise in the control room. The connection to the desks is provided by the PCoverIP technology, allowing up to four monitors per station. The same system is adopted for the 8 high-definition projectors, displaying critical information on the front wall of the control room.

3.3 Hierarchical structure with netbooted nodes

TDAQ decided to make extensive use of PC's with no Operating System (OS) on disk. Machines are booted via network using PXE. This is meant to remove the dependency from a hardware component that was expected to have high failure rate; more important for us, though, is the fact that it makes them easy to maintain over large numbers. The single image ensures complete uniformity, and no time is required to install the operating system - which is an important consideration for the fast replacement of some critical systems, such as ROSes, which cannot be redundant because of the direct connectivity to the front-ends.

As a downside, this boot scheme is not commonly used and not much supported by the Linux distributions, which means we need ad-hoc development and maintenance; furthermore, not every type of application can easily (if at all) be installed and configured.

For the distribution to all clients of the boot and OS images, and of the application software, we use a two level system of NFS servers, as shown in Figure 3.

The reference sources are stored on the central NAS. From there they are copied, using a hierarchical rsync mechanism, to all of the LFSes. In turn, each LFS provides boot services (DHCP and TFTP for PXE), and NFS sharing of OS and application software, to its allocated clients, which are usually located in the same rack.

In this hierarchical structure, the netbooted clients have a strong dependency on their LFS, and so they are not suitable for running critical services; the failure of an LFS can impact a significant

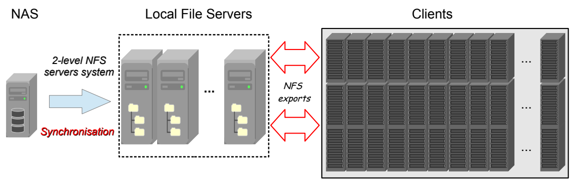

Figure 3: The OS and software are distributed from the NAS to the LFSes, which share them via NFS to the clients.

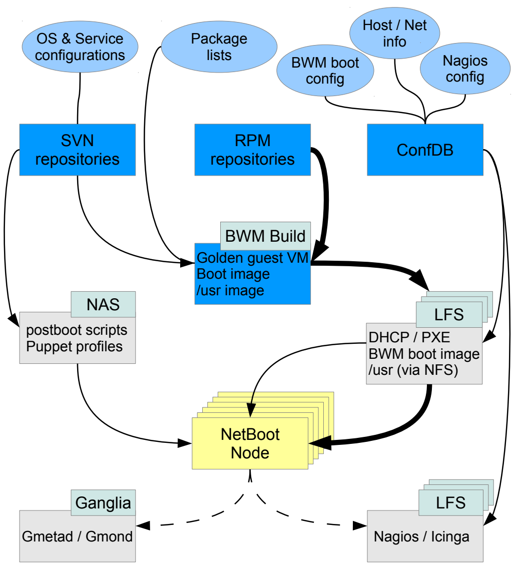

Figure 4: A NetBooted node receives OS and configuration information from various sources (thin arrows). Its boot and operating system are provided by BWM via its LFS (thick arrows). Monitoring information is collected by a Nagios server (usually the boot LFS); some hosts also send more detailed performance information to the dedicated Ganglia server (dashed arrows).

$(30 \sim 50)$ number of nodes. However the possibility of rapidly reassigning the clients from a failing LFS to another, combined with the large number of LFSes (~ 70), makes the structure sufficiently redundant and reliable for most uses.

The management of such a large number of netbooted nodes is far from trivial, so we developed a set of tools called the Configuration Database (ConfDB) [4]. Its main purpose is to collate data extracted from CERN's network database (LanDB) with our own configuration information, and generating configuration files on the LFSes for DHCP, PXE boot options, and for monitoring. It uses a python backend along with a MySQL database and a web based Graphical User Interface on the front end, and has greatly helped to speed up routine operations, such as registering and assigning clients to LFSes and even issuing of IPMI and system commands to multiple nodes in parallel. A flow chart of the information flow for this structure can be seen in Figure 4.

The netboot approach has been adopted for the majority of the data acquisition systems:

- Front-end systems: 159 ROSes, 183 VME Single Board Computers (SBCs), special-purpose nodes for subdetectors

- TDAQ Data flow: 69 machines
- TDAQ infrastructure and monitoring: 33 machines dedicated to DAQ control and 43 monitoring machines
- TDAQ High Level Trigger farm: 1656 machines

3.4 PreSeries

In addition to the main system there is the PreSeries system, a mostly isolated network where the Point 1 system is replicated at a small scale (for a total of 131 nodes), which is dedicated to release validation tests.

3.5 TestBed

The TDAQ TestBed architecture is also structured along the same lines, with a Linux-based Central File Server, four LFSes serving ∼130 netboot clients in the lab, and one LFS for external clients. To be able to provide more flexibility, more frequent updates and better AFS performance, the remaining clients are instead localboot systems: 18 "public" nodes, for develoment and general purpose, and 55 "worker" nodes, for software and network testing.

The testbed also hosts a setup for test and development of virtualization and storage, and a small scale setup for the Sim@P1 project discussed in Section 7.2.

4. Configuration Management

Given the large number of systems, and the considerable variety of their configurations, it is absolutely necessary for us to make sure that they all get installed correctly and that their configuration is kept consistent through their life cycle. In order to do this, we have adopted software tools for central configuration management [5]. An important additional benefit is that, combining these tools with an installation based on PXE and KickStart, the replacement of a system can be fully automated, and completed in a less than one hour.

4.1 Quattor and BWM post-boot (present)

For the systems in production as of early 2013, running SLC5, the primary tool used for configuration management is Quattor [6], the system commonly used at CERN. The Quattor system is composed of a server, which provides host profiles and is able to notify the clients when the profiles have changed, and of client-side agents, which enforce the configuration. The PAN compiler, executed on the server, transforms the host template definitions into XML profiles.

The SPMA component of Quattor provides very good control over the list of installed software packages, but requires considerable time and effort for the manual maintenance of package dependencies. Quattor provides other components for configuration of various system aspects, but unfortunately their implementations do not follow a coherent approach and many of them are not regularly maintained.

Another issue with Quattor is that it uses different languages between the templates, the functions inside the templates, and the client-side part of the component that is executed by the agent. This considerably steepens the learning curve for system administrators to be fully competent in its usage and development.

For the netbooted systems, we have been using our own in-house Boot-With-Me post-boot (BWM), a flexible system made of a hierarchy of shell scripts. A master script drives the execution of those scripts that correspond to host groups, sub-groups and finally a host-specific script, if present. This system has proven itself adequate to specialise the functionality of the netbooted single-image, but has a severe limitation in that it is only fully applicable at boot time, on the cleanly booted system. Also, scripting does not promote consistency in the configuration.

4.2 Puppet (future)

Because of the limitations and difficulties experienced with the above systems, we decided to look for a solution that would overcome those issues and provide a single, uniform configuration management for both local- and net-boot systems.

In 2010 we considered different options, including CFEngine3 [7] and Chef [8]. We finally decided to adopt Puppet [9] as it offers an approach strongly oriented to consistency and idempotence, a good arsenal of base types to cover common needs, a Domain-Specific Language that is easy to learn, and a lively and active user community. In 2012, CERN IT decided to start migrating from Quattor to Puppet. We started using Puppet for systems too complex to be effectively managed by Quattor, such as webservers.

For the netboot nodes, we have chosen to use the server-less mode of Puppet, to avoid any scalability issue. The configuration files are simply shared via NFS from our central NAS.

As of March 2013, the TDAQ TestBed is fully managed via Puppet. For Point 1, Puppet is being used in parallel with the legacy configuration tools on SLC5; the migration is ongoing, and expected to be completed during summer.

4.3 OS Repository Management

For OS Repository Management, we use snapshots of CERN's package repositories, which are the only repositories Point 1 is allowed to use. We have had sufficient functionality of it during 2012, allowing us to control what got upgraded and when, and theoretically rolled back if needed. We did however realise that we need a more flexible system, easier to manage, allowing for partial upgrades, client status reports, and so on. So we are considering the adoption of a repository management tool, possibly Pulp [10].

Even if we can perform rolling back, Puppet and Yum do not offer us the same detailed control as we used to have with Quattor and SPMA; we expect to invest into improving the package management functionality of Puppet later during the LS1.

5. Monitoring and alerting

Monitoring is a fundamental tool to proactively warn about any failure or degradation in the system in order to avoid or minimise any downtime. A monitoring system should be able to collect data about performance and health of the system, visualise them and eventually to alert those that need to be informed (by sms or mail).

Nowadays there are various monitoring packages available and the ones we currently using are Nagios [11] and Icinga [12] [13] integrated with Ganglia [14].

5.1 Nagios (present)

Nagios v2 is the tool adopted in 2007 to implement the monitoring of the ATLAS online farm. It has proven to be robust and reliable, but it has issues with scaling up with the size of the cluster. Indeed due to the large amount of checks and hosts (∼3000 hosts with up to ∼40 checks each) it was necessary to distribute the configuration and the work load over many servers (i.e. each LFS) in order to scale.

Since the number of servers, and nodes to be monitored by each server, is too large to be handled manually, the configuration files, describing the hosts and services to be checked, have to be generated automatically by ConfDB using a simplified and standardised structure and cannot take advantage of the full flexibility native to Nagios (e.g. host and service checks dependencies).

Eventually the data resulting from the checks is stored on a MySQL database which represents the current status; the database is hosted on a MySQL Cluster for performance and reliability reasons. The same data is also accumulated in RRD (Round-Robin Database, [15]) format and stored on our NAS, to be used later to produce graphs showing the historical evolution of the monitored parameters.

This implementation has the obvious disadvantage that the information is scattered across all the LFSes, while the standard Nagios GUI of each server can only display the data collected locally. To overcome this limitation, a web interface has been developed in-house to group all the information in a single page: the interface displays summaries and detailed data of the checks performed by each of the ∼ 80 servers, and provides, for each monitored host, quick links to the GUI of its Nagios server.

Further customisations have been introduced to adapt the existing plugins to suit our needs, and a full system has been put in place to monitor all the significant IPMI [16] sensor information (temperatures, fan and power supplies statuses etc.) of the PCs in the farm.

Despite the described disadvantages of the complex, custom distributed implementation, the system has been working well since the start-up of LHC operation in 2008.

5.2 Icinga/Ganglia (future)

The main goals in updating the current system are to simplify the current setup by reducing the number of distributed servers, while maintaining the same performances, to increase the flexibility of the configuration, and to substantially reduce the maintenance effort.

Icinga is a fork of Nagios v3 and is backward compatible: Nagios configurations, plugins and addons can all be used with Icinga. Though Icinga retains all the existing features of its predecessor, it builds on them to add many patches and features requested by the user community. Moreover a very active developer community provides regular major releases, bug fixes and new features. All these characteristics, and the backward compatibility with Nagios configurations, convinced us to test and eventually to adopt it.

As a first step to improve the monitoring system, we have recently introduced Ganglia to complement the information provided by Nagios and Icinga providing detailed performance information on interesting servers (e.g. LFS, virtual hosts, ...). Ganglia is a software package designed for monitoring the workload and performance of multiple large, and possibly geographically distributed, high performance computing clusters; contrary to Nagios and Icinga, it does not have advanced alerting capabilities.

We are already using Icinga in production in TestBed, on all hosts but with a subset of the checks. The tests of the integration with Ganglia are promising both in terms of performance and functionality. In Point 1 we also have Icinga in production for monitoring a small set of critical systems, and we have successfully performed scalability test.

For the full scale implementation of Icinga in Point 1 we will adopt Gearman [17] and mod_gearman [18], a queue system that allows the distribution of the load across many slave worker nodes, and therefore the scaling capability of Icinga.

6. Security, authentication and access restriction

The complexity of the ATLAS experiment motivated the deployment of an integrated Access Control System in order to guarantee safe and optimal access for a large number of users to the various software and hardware resources [19].

6.1 Access Manager

The Access Manager (AM) is the component, developed for the TDAQ system, used to implement the ATLAS policies that regulate access and operations by individual collaborators in the online computing system. It is responsible for authorising users, on the basis of their rights, whether or not to perform an action on the ATLAS online computing system during data taking. Its goal is to control operations executed by subjects in order to prevent actions that could damage or steal data and resources.

The hierarchical Role Based Access Control (RBAC) was the best choice as an underlying access control schema, as it fulfills the ATLAS protection requirements and offers the flexibility to accommodate the large number of users, roles and resources. The role, on the basis of policies defined, determines which resources can be accessed and permission is granted or denied according to the roles currently enabled for the user. A policy is a set of rules indicating which subjects are permitted access to which resources, using which actions, under which conditions.

Roles are defined in Lightweight Directory Access Protocol (LDAP) as Network Information Service (NIS) netgroups. This approach offers several advantages, including the ability to model a hierarchical structure and integration at the operating system / application level. The system currently consists of $\sim$ 360 unique roles and $\sim$ 4250 users.

The AM has been designed as a highly scalable system, capable of handling hundreds of requests in parallel and has been implemented using a client-server architecture: the client sends authorisation requests to the server and the server processes the requests and sends back the responses to the client.

7. Virtualisation

In recent years, virtualisation has been changing the landscape of computing. By allowing a single computer to run multiple operating systems, it allows it to consolidate multiple services and make better use of the ever increasing computing power of today's systems. When integrated with "cloud" management tools, it provides high flexibility in allocating, on demand, shares of computing power for specific applications.

In the context of the ATLAS TDAQ Farm, however, there is currently limited scope for virtualisation. The vast majority of the systems in the farm are dedicated to well defined applications, which are already capable of making full use of the computing and I/O capabilities of the hardware. Interposing additional virtualisation layer between these applications and the bare metal would only decrease their performance. Also, the TDAQ Partition tools already provide, at the application level above the OS, ways to redistribute tasks across the farm, negating the need for cloud management tools.

7.1 Current and future use

While the above considerations exclude most of the farm from the use of virtualisation, there are niche applications where it has proven useful in the ATLAS Online system.

The fact that virtualisation provides full isolation of multiple operating systems has been put to good use in the implementation of security-critical systems like the ATLAS Application Gateways, which provide controlled access to the ATCN. Multiple VM's provide redundancy, and they can be controlled and constrained by the containing hypervisor, which is not exposed to external access.

The other significant usage of virtualisation is for consolidation of critical core services with light computing and I/O load. This has allowed us to contain rack space occupation, while at the same time maintaining or improving the redundancy of the services. Virtualisation will also be used for the DCS systems that need to run applications only available under the Windows OS.

The hypervisor of choice for all these applications has been KVM [20] under SLC6.

7.2 Sim@P1 project

During the the LHC's LS1, when the accelerator and detectors will be undergoing maintenance and upgrade, the TDAQ Online farm will see limited usage of its primary task, the processing of live events. Its computing power could be used for other tasks, such as those usually undertaken by the Grid clusters participating in the ATLAS Distributed Computing - Monte Carlo simulations, reprocessing of ATLAS data etc. On the other hand, even during this period the ATCN must remain functional and secure to guarantee the control on the detector, and the possibility for subdetectors to carry out tests and calibrations during the maintenance work. The availability of virtualisation and cloud tools such as OpenStack [21] has allowed us to devise an implementation that will make the computing power available without any risk to ATCN.

8. Conclusions

After 3 years of the first LHC run, the LFS-based architecture seems to have a good and linear scalability. Over all we are happy with it, but it needs to be kept under control.

Netbooting might have limited functionality, but also has great flexibility in OS migration. Localboot used to be harder to configure, though programs like Puppet are changing this rapidly Our Monitoring systems will also see significant improvements during LS1.

Overall, the current architecture is sound and performing well. Changes to the existing system would only increase the complexity of it, without giving us any drastic improvements. So, we have decided to focus on trying to simplify and streamline the system, making it more robust and maintainable as a whole.

We expect that the experiences in cloud and virtualisation from the Sim@P1 project will be very useful for future evolutions.

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, "System Administration of ATLAS TDAQ Computing Environment", CHEP 2009

[4] ATLAS TDAQ SysAdmins, "Centralized configuration system for a large scale farm of network booted computers", J. Phys.: Conf. Ser. 396 (2012) 042004

[5] ATLAS TDAQ SysAdmins, "Upgrade and integration of the configuration and monitoring tools for the ATLAS Online farm", 2012 J. Phys.: Conf. Ser. 396 (2012) 042005

[6] Quattor: http://quattor.sourceforge.net/

[7] CFEngine: http://cfengine.com/

[8] Chef: http://www.opscode.com/chef/

[9] Puppet: http://puppetlabs.com/puppet/what-is-puppet/

[10] Pulp: http://www.pulpproject.org/

[11] Nagios: http://www.nagios.org

[12] Icinga: http://www.icinga.org

[13] ATLAS TDAQ SysAdmins, "Tools and strategies to monitor the ATLAS online computing farm", J. Phys.: Conf. Ser. 396 (2012) 042053

[14] Ganglia: http://ganglia.sourceforge.net

[15] RRDTool: http://oss.oetiker.ch/rrdtool/

[16] Intelligent Platform Management Interface (IPMI): http://www.intel.com/design/servers/ipmi/

[17] Gearman: http://gearman.org/

[18] mod_gearman: http://labs.consol.de/nagios/mod-gearman/

[19] ATLAS TDAQ SysAdmins, "Role Based Access Control system in the ATLAS experiment", CHEP 2010

[20] Kernel-based Virtual Machine: http://www.linux-kvm.org

[21] OpenStack cloud software: http://www.openstack.org/