Review of CERN Data Centre Infrastructure

P Andrade, T Bell, J van Eldik, G McCance, B Panzer-Steindel, M Coelho dos Santos, S Traylen and U Schwickerath

European Organization for Nuclear Research (CERN), 1211 Geneva 23, Switzerland

Tim.Bell@cern.ch

Abstract. The CERN Data Centre is reviewing strategies for optimizing the use of the existing infrastructure and expanding to a new data centre by studying how other large sites are being operated. Over the past six months, CERN has been investigating modern and widely-used tools and procedures used for virtualisation, clouds and fabric management in order to reduce operational effort, increase agility and support unattended remote data centres. This paper gives the details on the project's motivations, current status and areas for future investigation.

1. Introduction

When CERN was developing the computing infrastructure and facilities for the Tier-0 of the Worldwide LHC Computing Grid [1] early in 2000s, there were very limited tools for managing thousands of servers in a consistent fashion with a small support team. The approach of using a large number of commodity x86 based servers running Linux was gaining momentum but had not yet become mainstream. This led to the development of a number of sophisticated tools such as Quattor[2][3], LEMON[4], Secure INformation DElivery System (SINDES), SLS[5], Service Database (SDB) and State Management System (SMS).

Since that time, commercial companies such as Google, Amazon, Facebook and Yahoo! have dramatically passed the number of machines which CERN has to manage. While total server numbers are often confidential, Rackspace recently published they have over 79,000 machines in their centres[6]. Thus, the CERN computing infrastructure is no longer unique and is not on the leading edge for infrastructure deployments. The CERN requirements should therefore not be special and any functional requirements not met by leading tools deployed at other large scale centres should lead to a questioning of the need rather than development local solutions.

The Agile Infrastructure project at CERN was started to deploy the tools and processes for data centre management used elsewhere with minimal customisation. This paper reviews the strategy taken and the results so far.

2. Goals

The Agile Infrastructure activities are targeting a series of high level goals.

2.1. New data centre support

The Tier-0 data centre will be expanded in 2013 to include a second site at the Wigner Institute in Budapest, Hungary. This centre will provide up to 2.7MW of additional capacity (similar to the current capacity of the CERN Meyrin facility which currently houses around 8,000 servers). In addition, a fully remote centre provides potential benefits to support business continuity. The centre will be managed from CERN using a minimum of local ‘smart hands’ to perform installations, interventions and retire equipment.

The current tools would require significant rework to make them sufficiently robust and scalable to cover this additional capacity and method of working. Since the staff levels are expected to remain constant but the computing capacity increase substantially, efficiencies must be identified in the daily processes to avoid a reduction in service quality.

2.2. Sustainable tool maintenance and support
The data centre management tools have been built over the past 10 years and require substantial effort to maintain and perform mandatory changes (such as support for new operating systems or IPv6).

On top of the code maintenance itself, new service managers have to be trained in all the tools before they can be productive. As use of these tools outside of CERN is limited, they are unlikely to have the knowledge on arrival and require one-on-one coaching by an experienced engineer to become productive.

2.3. Improved user responsiveness
Moving towards a private cloud infrastructure allows many optimisations in IT processes. Self-service user kiosks can create virtual machines in minutes rather than waiting days for a physical server to be installed and allocated.

2.4. Enable cloud interfaces and architectures
Many experiments have already been testing cloud interfaces from their pilot factories. Using cloud APIs such as OCCI [7] or EC2 [8] along with libraries such as libcloud [9], these resources can be provided in a private cloud based on the experiment resource allocations.

Platform as a Service models can also give significant productivity gains by allowing users to select off-the-shelf standard configurations such as web application frameworks or databases, adding code and then connecting them to deliver the functionality desired. This compares with the current approach of defining a machine configuration from the bare metal to the application by hand.

2.5. Clearer understanding utilisation and monitoring
While tools such as LEMON give basic fabric monitoring information, identifying inefficiencies and bottlenecks requires more detailed data such as process accounting and network performance per stream. Extracting data from LEMON for analysis and cross correlation with other events is a very time consuming process which makes it difficult to identify the causes of inefficiency. Other data sources, such as grid CPU, network and disk utilisation, are in multiple data stores. Specialised programs are required to extract and match with fabric data such as CPU utilisation. The end result is that monitoring data cannot currently be exploited in a timely fashion.

2.6. More accurate and accessible accounting
Some services such as the CERN batch system provide precise accounting data which is reported and used for determining experiment usage. However, other parts of CERN infrastructure such as ad-hoc hardware requests, AFS space or the current CERN self-service virtualisation kiosk are currently not tracked in the same way. Thus, there is no single method to compare the experiments utilisation with the CERN pledges.

2.7. Improve efficiency over entire hardware lifetime
Current initial installation procedures, burn-in tests, hardware repair and retirement processes all contribute to time when a machine is out of production, using power but may be able to do useful work. Initial installations are in bulk so a small number of issues with a large delivery can lead to delays to the start of production. Burn in tests currently requires extensive engineer grade analysis to confirm compliance with specifications. Hardware repairs require approval of service managers and a

long delay to schedule the vendor intervention and return to production. Retirements involve bulk draining and idle hardware rather than running until the last minute.

3. Approach
To achieve these goals, we are following a similar model to many of the Internet service providers by adopting 'devops' principles [10] and a 'toolchain' approach [11].

3.1. CERN toolchain model
The tool chain for the CERN data centre is based on the Google toolchain model.

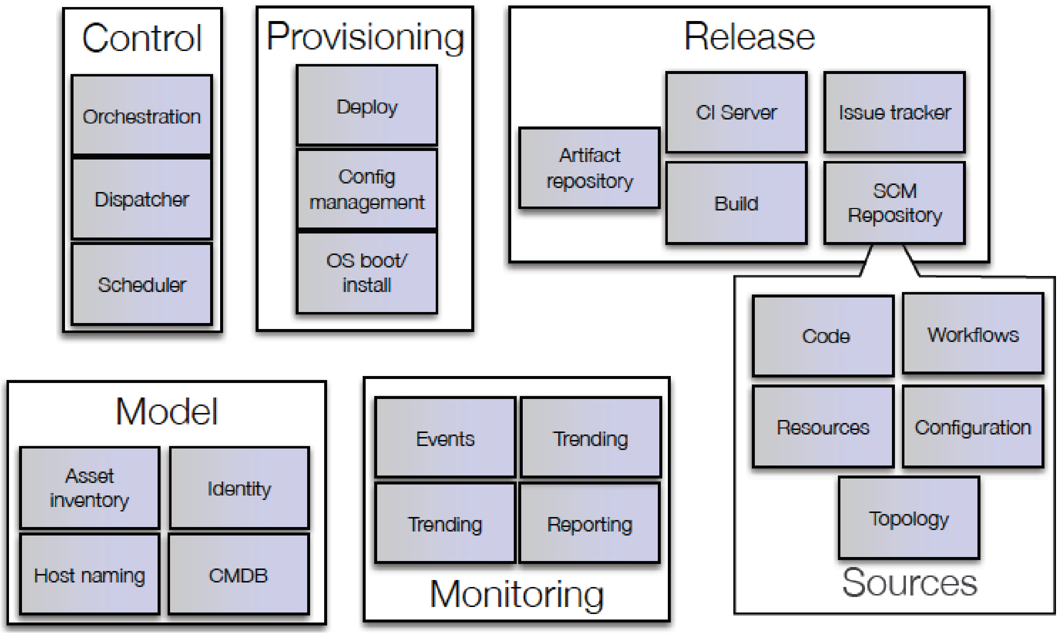

Figure 1 Google Toolchain Model

Given the large scope of the project, it is not possible or desirable to migrate all of our tools at once. Some tools, such as LANDB for network management, the CERN certificate authority and Active Directory for authentication, are sufficiently stable, functional and scalable to support the project. Others, such as the LEMON fabric monitoring tool, can be easily adapted and integrated with common open source tools.

In the remaining areas, an investigation of alternatives was made by discussing with other large scale data centre sites, reviewing internet sites and rapid prototyping. Particular attention was paid to the level of support from the community with respect to active code development, mailing lists, good software packaging and up to date documentation. From this, a set of new tools were chosen and connected to build a solution.

3.2. Facilities evolution
The Meyrin data centre is reaching the limits of its capabilities in view of the power and cooling available on the CERN site. Various solutions were studied including building an additional centre on the Prevessin site but these proved to be both expensive and requiring highly specialised expertise. An alternative approach was followed in 2011 to issue a call for tender to the CERN member states for remote hosting of CERN equipment, along with local support such as hardware repair, installation and cabling services. The Wigner Data Centre in Budapest offer was accepted following analysis of the 16 bids from 9 countries.

The new facility is due to be available by the end of 2012 with 725 m^2 of space and 2.7MWatts maximum capacity available to CERN. Densities of an average of 10kW/rack are supported along with 2 independent high voltage lines to the site. Full UPS coverage is available for all servers with N+1 cooling redundancy.

During 2012, the 100Gbit/s network connectivity will be set up and test servers installed. At the beginning of 2013 100kW of equipment will be installed to test out the various operation procedures between CERN and Wigner. Towards the end of 2013, the capacity will expand to 600kW using the Infrastructure-as-a-Service tools described below to be in full production at the beginning of 2014.

3.3. Component and stores catalog

The software, interfaces and data stores are documented in a component and stores catalog [12]. This is to address previous problems where ownership of software was not clear or the impact of a change of data structures on tools was not known.

Entries are created for each set of fields such as which machines are in a host group or are part of the same maintenance contract. For each of these stores, there is a master store defined (‘Source of Truth’) and associated cache stores which are responsible for ensuring that their data is maintained up to date within a permitted time window to ensure eventual consistency. One of the major deficiencies of the previous system was the number of “cache synchronizations” that needed to be performed.

Component cards are created for all tools used within the project. These cards define the implementation language, responsibility and the data stores used. These cards are created for both externally developed and internally developed tools.

Using this structure, it will be possible to rapidly determine which tools will be affected by a change of schema and what the effort would be to replace one tool by another. A uniform data model and access library is being provided to help ensure consistency of retrieval and update.

3.4. Hardware installation and management

In this area we are addressing three sub-areas where pain points and possible improvements have been identified.

3.4.1. *Server registration.* Arrival of new servers and the preparation for burn-in currently requires a substantial chain of human actions to bootstrap a server into an initial operating system installation and configuration with network access. The amount of manual work will be significantly reduced by having a system that allows servers to register themselves into the different databases required. At the same time, this will make the process more robust both by eliminating human error in the mentioned registration and also by a higher tolerance to unexpected variations among the servers and deployment infrastructure such as network cabling and disk setup.

3.4.2. Server burn-in. New servers are put through a burn-in test on delivery. Along with identifying any broken hardware components, this has helped for early identification of severe hardware problems such as memory issues (ECC errors), storage issues (disk firmware problems, vibrations), internal temperature problems and other firmware problems (BIOS, BMC), etc.

The detection of these problems is achieved by running a set of different tests across the different components. Processor (burnP6, burnK7, burnMMX), disk (badblocks, S.M.A.R.T. self-test, fsprobe), RAM (memtester, fsprobe) and network (netperf) are stressed for several days to identify potentially early fail components.

The other requirements are also validated during burn-in such as remote serial console, IPMI remote power control and adequate hardware monitoring (S.M.A.R.T. counters, SCSI Enclosure Services). Performance measurements specified in the server tender documents are also repeated, including CPU performance (HEPSPEC06), disk performance (FIO) and power measurement. The results are then recorded.

Server burn-in will occur automatically after server registration and will run on the operating system to be used for the production environment, including production hardware monitoring.

Finally, until the commercial acceptance of the batch of servers, they can be used within the Infrastructure-as-a-Service layer to run guests with lower quality of service and hardware errors monitored closely.

3.4.3. Server Hardware Inventory. In this area, we plan to record not only what the server hardware configuration should be, but also its current state and the history of changes. As an example, knowing how many disks a server should have, how many disks a server does have and what disks were replaced, and when.

With this knowledge, the servers can be used in a more resilient way. Services can be made tolerant of some hardware failures and the machines can run in a degraded mode while still being available for remote debugging.

An accurate inventory of servers and their components (disk drives, memory modules, etc) as well as a systematic recording of changes over the servers' lifecycle enables better failure tracking and trending of different components.

3.5. Infrastructure as a Service

This area will provide a private cloud with interfaces that allow creation of virtual machines by customer groups and end users, to cater for a range of use cases from the CERN Virtual Infrastructure (CVI) to Lxcloud services. Although these use cases differ in their Quality of Service needs, they can use a single, multi-site IaaS to satisfy a full spectrum of reliability and dynamic resourcing services.

The service consolidation needs address the requirement for production services such as experiment specific 'VOBoxes', custom application servers, personal virtual workstations and test servers for a variety of operating systems. They hold valuable data and contain state information which has to be preserved for the life time of the virtual machine (months to years). From the current experiences with virtualisation at CERN, an equal split of Linux and Windows would be expected. Current growth rates would predict around 10,000 VMs by 2015. This style of server can be viewed as a "pet" [13], as they have meaningful names, are valuable to their owners and worthy of investing substantial effort to recover the server in the event of problems.

The private cloud use case has been explored in the Lxcloud project. These VMs are predominantly Linux based and will often be instantiated by automated procedures such as experiment job factories or scale out cloud orchestrators. By 2015, the number of VMs in this class could grow to around 300,000 (although this is very dependent on the number of cores to be allocated per VM). The VMs can be viewed as "cattle", they are given numbers and re-installation is the easiest way of recovering from failure.

The deployment uses four of the core components of the OpenStack project [14]

1. Nova is used to provision and manage large networks of virtual machines, creating a redundant and scalable cloud computing platform. Nova provides two APIs (OpenStack Nova and EC2) to orchestrate a cloud, including running instances and managing networks.
2. Keystone provides Identity, Token, Catalog and Policy services for use specifically by projects in the OpenStack family. It supports multiple backend data stores and work is on-going to integrate with Active Directory, as used at CERN.
3. Glance is an image service providing discovery, registration, and delivery services for virtual disk images, that can be stored in a variety of back-end stores. Glance allows uploads of private and public images in a variety of formats, including qcow2 (KVM) and VHD (Hyper-V). Image distribution and contextualisation is being adapted to the interfaces defined by the HEPiX virtualisation working group.
4. Horizon gives the OpenStack Dashboard which is a web interface to enable administrators and users to access and provision cloud-based resources through a self-service portal. The core reference implementation comes from OpenStack but it can be extended using Django to include site specific functionality.

Our deployment of OpenStack Compute will integrate with various services already in widespread use in CERN.

- LDAP / Active Directory for user and tenant information in Keystone
- Database on Demand service for MySQL databases required by Nova and Glance
- Messaging services, used by Nova
- The CERN LANDB for network management by Nova

The first of the pre-production services was delivered for testing at the end of May. This release is based on the Essex version of OpenStack and is deployed using the Fedora 16 operating system on the KVM hypervisors. System installation and configuration is achieved using the OpenStack Puppet modules [15] as provided by PuppetLabs [16]. The release is used to test OpenStack features such as virtual machine migration, block storage and changes of flavours (such as adding more virtual cores and memory to a live system). The configuration supports around 300 virtual machines.

Future releases will test deployments with advanced network configurations, virtual machine backups, Windows guests and scaling tests. The hypervisor operating system will be migrated from Fedora 16 to Scientific Linux CERN 6. A production quality Infrastructure-as-a-Service is targeted for deployment by the end of 2012.

3.6. Configuration management and tools

One of the major goals of the project is to leverage the operational tools experience built up by the Open Source community over the last few years. The software ecosystem for configuration and automation is constantly evolving and being improved by the community – when a new tool comes out and gains traction, we typically find that the integration with other common components is already done and is maintained for us. As a member of these "devops" communities, we will also contribute, returning fixes, features and developments rather than developing our own internal solutions.

Due to the incremental way in which the current CERN toolset was developed, there is a significant lack of "dynamicness" in the whole system. The interactions between the components have "grown up" and "grown over" – they are suboptimal and, in some cases, no longer clearly understood. This has several aspects:

- The CDB configuration template system is very static, text-based and does not lend itself well to automation. A single configuration change to a cluster can take a lot of searching and updates can take between 2 and 20 minutes to commit into the system, depending on the activities of other admins.
- The overgrown architecture with lack of clear "sources of truth" means that different parts of the system need to store the same information. This typically results in procedural inefficiency, such as administrators needing to wait for 30 minutes for data stores to synchronise before moving to the next step in the workflow.
- There is no single security solution for these tools, as the security has been evolved over the years and differently, per component – this effectively bars much of the automation that could otherwise be done. There are many potentially automatable procedures and workflows where the sole "added-value" of the human being is to enter their password.
- The maintenance cost of the system as a whole is rather high. Furthermore, the costs of taking a "standard solution" and integrating it are also quite high – this typically results in off-the-shelf solutions being rejected in favour of more in-house developments, which since they are being developed from scratch, are easier to integrate with the existing system.

In order to clean up the architecture and ensure we maximize the use of external tools, we opted to replace the entire configuration infrastructure rather than migrate component by component. The strategy is to set up a new system from scratch, understand it with pre-production services, and then slowly migrate the current services over to the new system as it matures. The new data centre in

Hungary will only use the new configuration tools, and provides a hard deadline for the project to ensure the focus on production.

The requirements for the new system are:

- It needs to scale up to the number of physical and virtual nodes we will have in the new system. Depending on how many cores per VM, we expect to have up to 300,000 guests in the next 3 years.
- It needs to scale up in terms of admins and distinct clusters. At CERN, we have over 100 distinct functional “clusters” that need to be managed by separate admin teams. Although we have some very large clusters (such as the disk servers and the batch compute service), over half of the clusters currently managed have less than 10 nodes in them. There are additionally over 700 other Linux service nodes that are “hand-managed” by admins around the site that it would desirable to bring into the new configuration system
- It needs to be sufficiently dynamic to allow virtual nodes to “come and go” as necessary, with no human intervention.

The new system is based around the Puppet configuration tool, which was chosen from the two leading contenders in the configuration arena, Puppet and Chef [17]. Puppet is a declarative system – the templates declare the working state of the system, and the Puppet “runs” bring the system in line with this. Chef is an imperative system (it aims to declare the required changes to the system). Both tools are mature, well integrated with other developments, and have a good, active community behind them. The strict declarative approach of Puppet was felt to be a better fit for our way of working and for the large number of distinct administrative groups that the tool needs to support.

1. At the centre of the system is the “Puppet master”, which is a horizontally scalable Ruby on Rails application. Every node in the system calls in periodically, and starts a “Puppet run”: during the Puppet run, relevant “facts” of information (OS, hardware, etc.) are determined on the node itself and passed to the compilation which compiles the admin-supplied templates and generates a set of actions on the box to bring it configuration in line with the desired state. Nothing is done if the box already has the desired configuration.
2. The admin-supplied templates themselves are stored in an external git repository. Although the currently supported version control system at CERN is Subversion, we chose git since this is used by the overwhelming majority of the Puppet community. Many of the procedures and tools that we plan to use assume git as the basis of your template management. The use of git allows different branches (e.g. one-off feature, testing, pre-production, production) to be easily handled. git will be used together with a code-review tool (to be decided) to allow changes to be efficiently peer-reviewed before being deployed into production.
3. Multi-host “actions” and “queries” on the nodes can be applied using the “mcollective” tool. This uses a messaging system to “broadcast” the required action to the nodes, which then run it, and send back the results to the client. The results can be made available programmatically (via JSON) or rendered for output by the mcollective client command line tool. For the messaging system, we currently use Apollo [18], the commercially supported fork of Apache ActiveMQ.
4. A “dashboard” interface and node classifier is supplied via the “Foreman” tool [19] which defines clusters of similar machine configurations and allows hosts to be moved between them. It is accessible via a web interface or a REST API for automation. The Puppet compilation reports from each node are sent to Foreman, allowing cluster administrators to see the current state of their cluster and follow up any errors.
5. The software release chain for our own components makes use of “git” or “Subversion” for the version control, Fedora Koji [20] for the automated build, and mash to provide the yum repository management. The boxes themselves obtain all their software via the normal “yum” tool. Continuous integration will be added later.
6. X.509 host certificates are normally provided by the puppet master (which maintains its own internal certificate authority). We have an in-house development which instead allows to

Puppet master to make use of the CERN certificate authority for providing the X.509 host certificates. This also enables scale-out implementations which have multiple puppet masters and a single certificate authority.

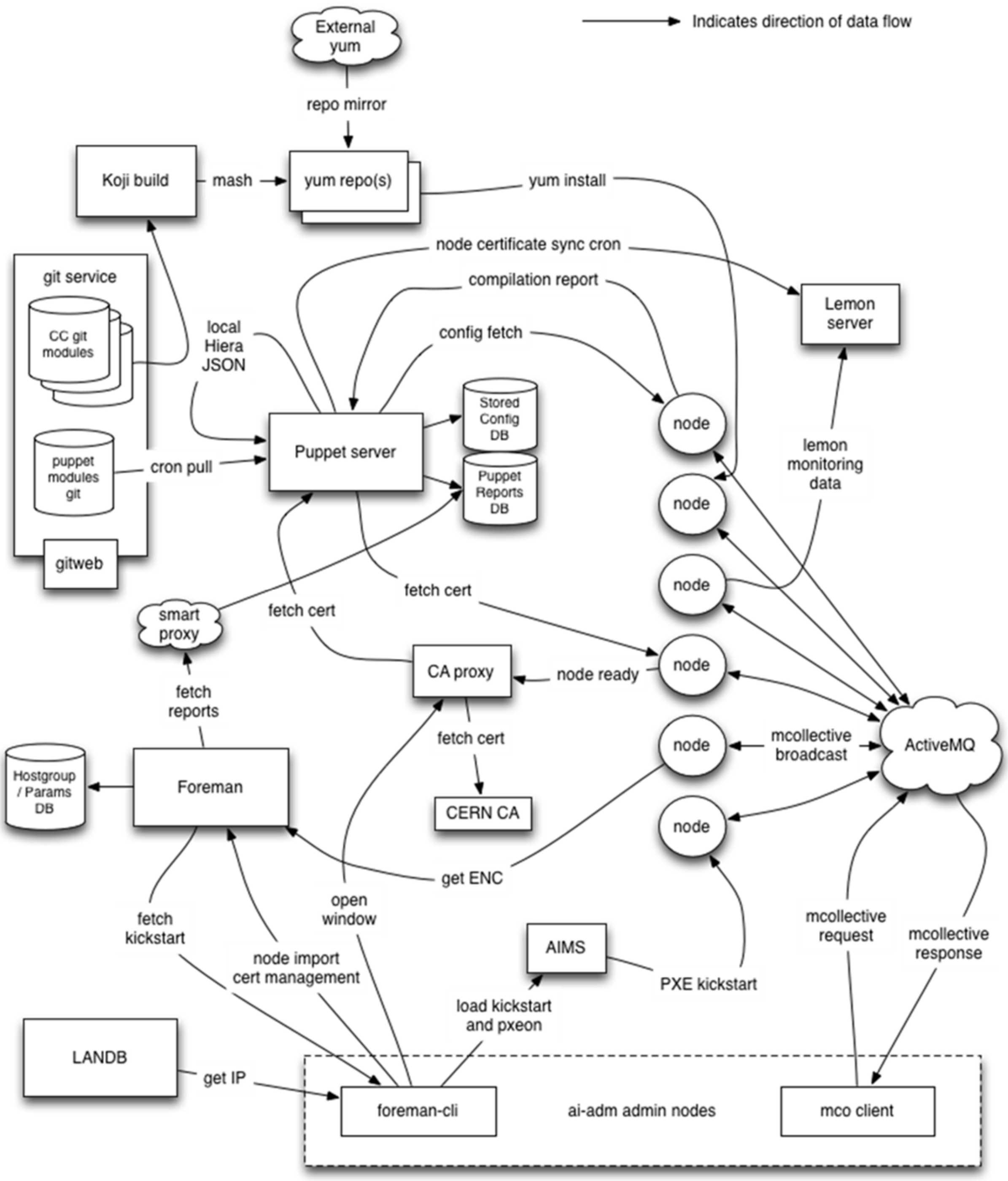

Figure 2 Configuration Management Data Flow

The current implementation is already providing more automation:

- Host moves and host interventions formerly required manual steps to update the state as the workflow progressed (such as moving hosts between clusters). The new interfaces are machine addressable, so are much more amenable to automation.
- Now the build chain is deployed, rollout of feature fixes can be automated – with deployment on “test” boxes, and automated “progression” of the feature fixes to production as a result of configuration code review.

The current prototype system has all of the components described above – it is current supporting 200 hosts in pre-production mode. The focus at the moment is to get the system stable and rewrite our operational procedures for the new tools, addressing the current inefficiencies. The infrastructure will soon be expanded to support a number of pre-production CERN services.

3.7. Scheduling and Accounting

As mentioned above, the new infrastructure should be able to support ~300,000 virtual machines. The hardware will be distributed over two geographically separated data centres, offering new possibilities for fail-over and business continuity models compared to today. On the other hand, access to CERN computing resources should be transparent regardless of their location, which implies one and only one entry point. For the users, the two centres should look like a single resource, and belong to one single large infrastructure which is capable to support all use cases at CERN.

This has implications for the Infrastructure-as-a-Service topology, with the approach below addressing these requirements:

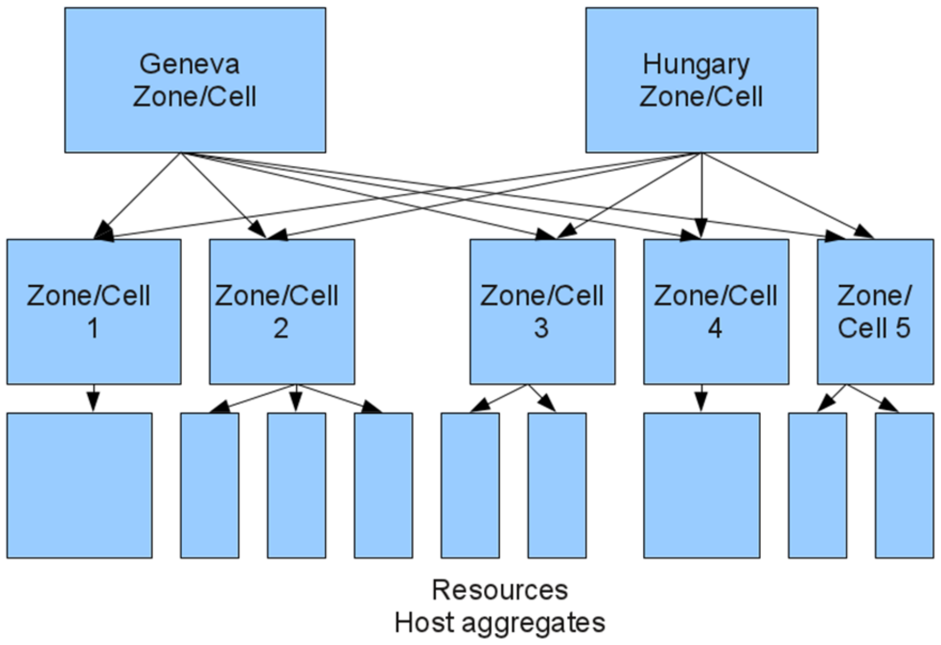

Figure 3 Infrastructure Topology for multi-centre deployment

Each physically separated region has an IaaS entry point which knows about all zones and cells at all sites, and each zone has two parent instances. This way, the two entry points are equivalent, and can be hidden behind a (load balanced) DNS alias.

For business continuity and an efficient fail-over mechanism, it is still necessary that users can require their machines to be dispatched in a specific data centre. This is needed so that critical services are distributed over both centres for high-availability purposes.

A key design decision is that resources and applications are separated from each other. As an example, the resources for the batch farm run on top of the IaaS layer, and the batch service itself does not need to know whether it is running on physical or virtual machines. As a result, the migration to the new schema will be entirely transparent to the users. The batch service managers become users of the infrastructure in the same way as other users requesting servers. The batch service gets a quota of resources assigned, and redistributes these resources to the end-users by instantiating batch nodes and adding them to the batch farm.

The infrastructure as a service layer is generic and can be used by all applications. This is required to allow for efficient use of resources by sharing them amongst all applications. Some grouping of hardware will be required as virtual machines which run on hardware that offers a better quality of service by having redundant power supplies, better network connectivity, UPS coverage or hardware RAID should be considered differently compared to virtual machines for number crunching applications, which can run on less expensive off-the-shelf hardware. Users should be able to pass the information to the system to which category their virtual machine belongs.

Resource allocations need to be done centrally. Cloud solutions support fixed quotas per user community, e.g. based on the CPU resources allocated for each community. Within this quota, the resource managers for the user communities decide on how they want to exploit these resources. As an example, an experiment could decide to only use batch resources, in which case their whole allocation is transferred into a share value on the public batch farm. Another user community may run a build farm and temporarily increase it by reducing their batch farm share. In this case, a mechanism should automatically increase their quota for their build farm (and launch additional VMs), and reduce their share on the batch farm. In the background these activities should trigger a migration of resources from the batch farm to the build farm of the experiment. A resource manager is needed to coordinate such activities. A joint project called “Cloudman”[21] between CERN and BARC is aiming to solve this problem.

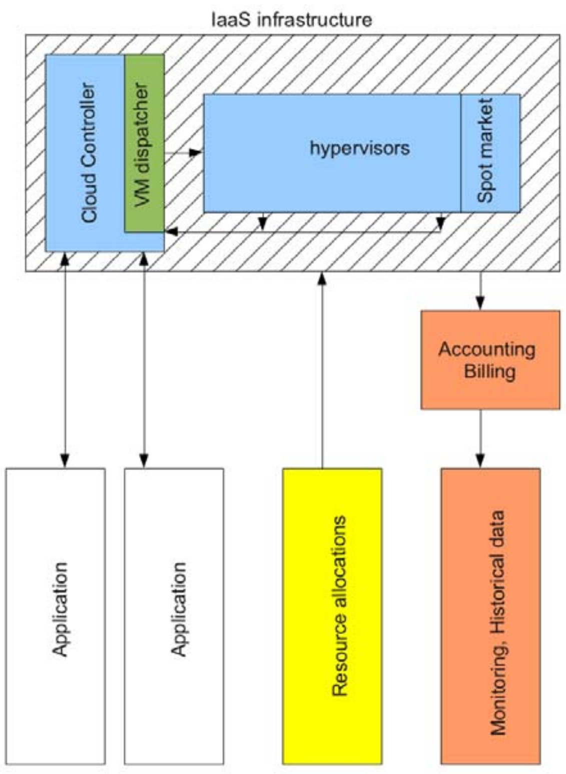

Figure 4 Cloudman High Level Design

The model implemented by static resources allocations closely emulates the current situation in HEP with physical machines. Investigations are ongoing to further improve the centre’s efficiency by a spot market where user communities can take advantage of unallocated or under-used resources which could be of a lower quality of service (such as termination at short notice).

Along with quota, accounting and billing features are available within IaaS implementations and can be used to track usage of the resources by the user communities. The information can be further enhanced by detailed monitoring data from the hypervisors. It is therefore not necessary to add accounting functionality into the virtual machine images themselves.

3.8. Monitoring

Infrastructure monitoring is a core activity in the daily operation of any data centre. For a successful execution of monitoring activities it is important to continuously monitor the status of all resources (network equipment, physical machines, virtual machines, operating systems, applications services, data stores, process, users, etc.), to process monitoring data and promptly deliver monitoring results (notifications, alarms, reports, etc.) to the appropriate target, and to allow the execution of complex queries across distinct monitoring data.

Taking in consideration these motivations, the project is trying to improve how monitoring is being performed. Today, multiple monitoring applications are in place, often tailored for monitoring the status of specific resources. These are independent tools based on different tool chains. Despite this heterogeneity, they all share a similar architecture and all face the same limitations, which leads to unnecessary duplication of effort and to increased difficulties in sharing monitoring data. To improve this situation the project is working towards the definition and implementation of a common monitoring architecture. This common vision will promote the sharing of monitoring components and allow the execution of more complex queries bridging different monitoring data sets.

To make sure an improved monitoring solution is delivered, a number of constraints, related to the technology and data, were identified to steer the discussion towards a final architecture. From the data perspective the monitoring architecture should aggregate all monitoring data in a large data store to allow advanced analysis tasks, and be based on a simple and well-supported data format to allow monitoring data to be easily accessed. From the technology perspective, the monitoring architecture should follow a tool chain approach where each component can be easily replaced by a better one, and provide well established solutions for each layer of the architecture by adopting existing tools and avoid home grown solutions whenever possible.

The following architecture has been identified as the base for all monitoring applications in the data centre.

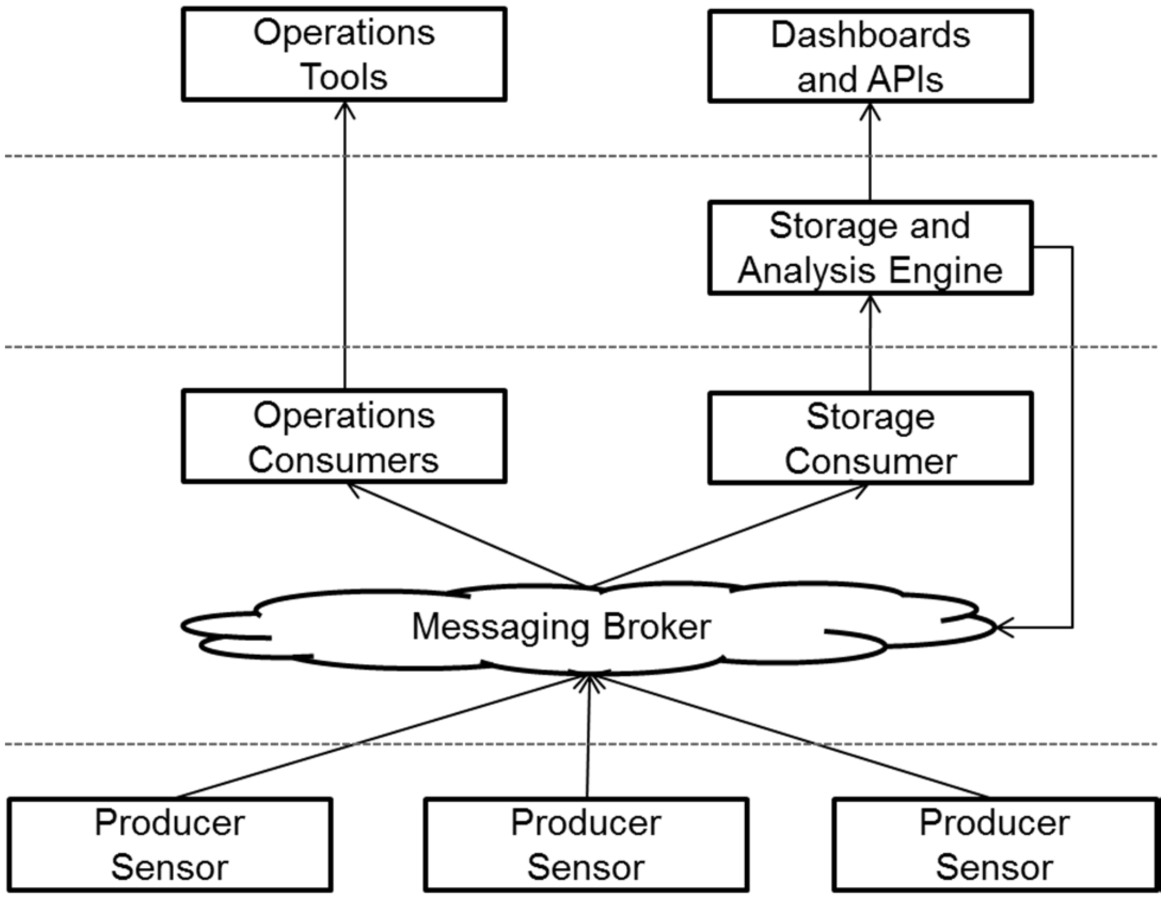

Figure 5 Monitoring Architecture

Data is produced by multiple sensors distributed across the data centre, data is transported through a messaging infrastructure allowing aggregation, data is stored and analysed in a large data store, and finally data is made available via tools, dashboards, and APIs tailored for operations and for analysis/historical queries.

At the core of the architecture is the data format. All monitoring messages will be formatted in JSON [22]. The usage of this lightweight data-interchange format enables a simple and powerful mechanism to seamlessly handle all monitoring messages across the infrastructure.

All data centre resources generate monitoring data. Sensors/producers running locally in each resource publish their monitoring data, based on the defined data format, to the messaging infrastructure. To abstract from the selected messaging broker all monitoring sensors should use common messaging libraries. It should be mentioned that, if required for performance reasons, monitoring data might also be pre-processed at each resource before being published or be pre-aggregated before arriving to the messaging infrastructure. Monitoring data coming from closed monitoring producers must also be integrated by injecting the final results into the messaging layer or by exporting relevant data at an intermediate stage.

The transport of monitoring data is based on a messaging infrastructure. The core of this infrastructure is the message brokers, which supports multiple deployment configurations such as broker load balancing, brokers hierarchy and brokers network. To identify the most appropriate deployment scenario, the requirements of each monitoring application must be clearly analyzed taking in consideration various aspects, such as, the total number of producers and consumers, the size of each monitoring message, the rate of the monitoring messages, etc. For a first implementation the selected technology was Apollo. This choice was motivated by the prior positive experience in IT and the experiments.

One of the architecture constraints already introduced defines the need to store of all monitoring data in a large data store following a big data approach. Such store makes easier not only the sharing of the monitoring data itself but also the sharing of analysis tools from different applications. Having a central store also allows the possibility of re-feeding the store with data produced by analysis tools and consequently executes advance queries on raw data and curated data at the same time. For a first implementation the selected technology is Hadoop and the map-reduce framework [24][25]. The map-reduce paradigm is a good match for the monitoring use cases and it has been used successfully at scale by other large projects worldwide. Hadoop also offers a wide variety of related modules, such as Hive and Pig, which are useful add-ons for some monitoring applications where a different programming abstraction is required [26].

Finally, the monitoring architecture defines different visualization tools, to provide the appropriate information to the correct target. Real-time operations notifications should be delivered as fast as possible to the appropriate responsible to guarantee that correct actions are taken. In addition, powerful dashboards and APIs must also be made available allowing an efficient and simple display/query of all monitoring data (raw data, curated data, and historical data). For the first implementation, the selected technology is Splunk [27] due to its powerful engine to combine data and the simplicity to create advance dashboards.

4. Status and Timelines

Along with the new data centre availability at the start of 2013, the long shutdown of the LHC during 2013/2014 provides a window for significant change. While much of the change outlined above can be performed in a transparent way to maintain compatibility for legacy applications, there is also potential for new applications to take advantage of the additional functionality to move to an alternative architecture based on an infrastructure as a service model and modern configuration management and monitoring.

It is expected that new services will be able to start using this infrastructure by the end of 2012, migration of old services will be performed during 2013 and the current toolset can be stopped in 2014.

5. Conclusions

The CERN computing facilities, tools and processes are undergoing a major transformation to benefit from the additional data centre capacity while maintaining the number of staff at the current levels. By using Open Source software with a toolchain methodology, this can be achieved within a much shorter time window and is more sustainable in the long term compared to the current home-grown tools. Initial results are positive based on Puppet, OpenStack, ActiveMQ and Hadoop as key components of the CERN toolchain.

References

[1] LCG - Worldwide LHC Computing Grid, http://lcg.web.cern.ch/LCG/
[2] Poznanski P. and Melia, G. C. and Leiva, R. G. and Cons, L., Quattor - a framework for managing grid-enabled large scale computing fabrics, In Memorias de la XXX Cconferencia Latinoamericana de Inform'atica, pp 777-789, 2004
[3] http://quattor.sourceforge.net/
[4] Babik Marian et al, LEMON - LHC Era Monitoring for Large-Scale Infrastructures, 2011 J. Phys.: Conf. Ser. 331 052025
[5] Sebastian Lopienski - Service level status—a new real-time status display for IT services, 2008 J. Phys.: Conf. Ser. 119 052025
[6] Rackspace Press Release, Financial Results 2011, http://cern.ch/go/KFS7
[7] Open Cloud Computing Interface http://occi-wg.org/

[8] Amazon Elastic Compute Cloud (EC2) URL http://aws.amazon.com/ec2
[9] Apache Libcloud API http://libcloud.apache.org/
[10] Rajiv Pant – Devops, http://www.rajiv.com/blog/2009/03/17/technology-department/
[11] Damon Edwards, DTO Solutions - Toolchain Approach, http://cern.ch/go/8NbJ
[12] Agile Catalog, http://agilecat.web.cern.ch/
[13] Pets and Cattle http://www.slideshare.net/randybias/architectures-for-open-and-scalable-clouds
[14] OpenStack, http://openstack.org
[15] OpenStack Puppet Modules, http://github.com/puppetlabs/puppetlabs-openstack
[16] Puppetlabs, http://puppetlabs.com/
[17] Opscode Chef, http://wiki.opscode.com/display/chef/Home
[18] Apollo, ActiveMQ next generation http://activemq.apache.org/apollo/
[19] Foreman, http://theforeman.org/
[20] Koji, http://fedoraproject.org/wiki/Koji
[21] Cloudman, http://cern.ch/go/G9pg
[22] Javascript Object Notation http://www.json.org/
[23] Challenges and Opportunities with Big Data, Cloud Computing Journal
[24] http://hadoop.apache.org/
[25] MapReduce: simplified data processing on large clusters, http://dl.acm.org/citation.cfm?id=1327492
[26] Comparing high level mapreduce query languages, http://dl.acm.org/citation.cfm?id=2042527
[27] Splunk http://www.splunk.com/