# Quattor: Tools and Techniques for the Configuration, Installation and Management of Large-Scale Grid Computing Fabrics

R. García Leiva
*Departament of Theoretical Physics, Universidad Autónoma de Madrid, Ctra Colmenar Km 15, 28049 Madrid, Spain*

M. Barroso López, G. Cancio Meliá, B. Chardi Marco, L. Cons and P. Poznański
*CERN, CH1211 Geneva-23, Switzerland*

A. Washbrook
*Department of Physics, The University of Liverpool, Liverpool L69 7ZE, UK*

E. Ferro
*INFN-LNL, Viale dell Universita 2, I-35020 Legnaro (PADOVA), Italy*

A. Holt
*University of Edinburgh, Old College, South Bridge, Edinburgh EH8 9YL, UK*

April 1, 2004

**Abstract.** This paper describes the quattor tool suite, a new system for the installation, configuration and management of operating systems and application software for computing fabrics. At present Unix derivatives such as Linux and Solaris are supported. Quattor is a powerful, portable and modular open source solution that has been shown to scale to thousands of computing nodes and offers a significant reduction in management costs for large computing fabrics. The quattor tool suite includes innovations compared to existing solutions which make it very useful for computing fabrics integrated into grid environments. Evaluations of the tool suite in current large scale computing environments are presented.

**Keywords:** quattor, fabric management, installation and configuration, clusters, Unix system management.

## 1. Introduction

*Quattor*[1] is a new tool suite for the installation, configuration and management of operating systems and application software for computing fabrics. Quattor is a powerful, portable and modular open source solution that scales to thousands of nodes, offering a significant reduction in management costs for large computing fabrics. Work on quattor started within the scope of the EU DataGrid project (Jan. 2001 - Dec. 2003), but development and maintenance are now coordinated by CERN (IT department), in collaboration with other partner institutes. For more

© 2004 *Kluwer Academic Publishers. Printed in the Netherlands.*

information about quattor, and to download the latest software release, please refer to the quattor web page at `http://quattor.org`.

DataGrid was a three-year UE funded project with the objective to develop the next generation computing infrastructure for data intensive grid applications. The project was organized into work packages: middleware development, grid testbed infrastructure, scientific applications, dissemination and management.

The goal of Work Package number 4 (WP4) was to provide all the necessary tools to manage a computing fabric providing grid services on clusters of thousands of nodes (Bärring et al, 2003). The tasks of the work package were divided into two areas: user job management (grid and local users), and automated management of large grid computing fabrics. The quattor tool suite project is one of the key results from these WP4 developments.

This paper describes the architecture of the quattor tool suite, and the actual experience of using it for the automated management of large computing fabrics. The rest of this section briefly reviews the challenges related to fabric management. Section 2 describes the architecture behind quattor. Section 3 shows the results achieved so far with quattor. And finally, Section 4 sums up the experience and current status, and describes the planed future enhancements to the quattor tool suite.

### 1.1. Management of Grid Computing Fabrics

A typical fabric for grid computing consists of clusters of computing nodes, where user jobs are run, and a number of grid and network infrastructure components:

- master nodes which co-ordinate the computing node clusters for batch and interactive services,
- storage servers such as disk servers (e.g. NFS, RFIO, GridFTP), and tape infrastructure (tape servers and robotics),
- installation and software repository servers,
- information servers (e.g. database and monitoring servers),
- network infrastructure (switches, DNS servers), and
- miscellaneous servers (time servers, authentication and authorization servers).

Managing a computer fabric with these components not only involves the management of services individually but also the management of

the dependencies between services and the consequent ordering between operations that have to be performed (Cancio et al, 2001). This latter complication is very important. For instance, upgrading the kernel version of a computing node involves the installation of the software package with the new kernel, and a modification in the configuration of the kernel loader (for example grub loader). This quite trivial example already exposes the importance of the order in which the operations have to be performed: in this case the configuration of the kernel loader has to be changed after the installation of the new kernel.

A further complication is that some operations can take a significant amount of time. In the example above, if there are user jobs running on the computing node, then the configuration change has to wait until all the jobs have finished. This operation may take days on some nodes whereas it may be immediate on others (those without running jobs). Such long time spans increase the risk of queuing up several operations on some nodes, for instance an OS upgrade might be requested for the following day on some of the node clients in the example above (Cancio et al, 2001).

Currently these types of operations are mostly performed manually, which can be error-prone, as well as being difficult and expensive to manage in large cluster environments. There is demand for a tool suite that allows for a significantly increased automation of those operations while still leaving the overall control and supervision of the operations in the hands of the expert system administrator. It is also important that the automation takes into account the integrity of user jobs. Specifically, fabric management operations should not be allowed to abort a user job nor its run-time environment unless essential intervention is required, for example an emergency due to a security incident.

The experience in DataGrid during these 3 years has also shown that there is a real need to be able to install, configure and manage the small, medium and large sites providing grid services. The proper integration of these sites into production grids of ever increasing size and complexity necessarily requires well managed and stable computing fabrics. Wrong or faulty grid services configurations not only affect the functioning of the site, but the performance of the whole grid (for example, a missconfigured bath system can attract all the grid resource requests and collapse the grid job submission).

The size of a site is an important complexity dimension in cluster environments, but the fact that a cluster is grid enabled adds another complexity dimension that is independent of the size, it affects equally to small, medium and large sites. This additional level of complexity is caused by the grid middleware configuration, the grid policies, and so on. This complexity factor is also solved by automated fabric manage-

ment tools, and its solution is critical for the manageability, stability and security of grid enabled computing fabrics.

### 1.2. Tools for System Administration

There are many tools for the installation, configuration and management of a network of computers already available. Please refer to (Barroso et al, 2001) for an evaluation of the tools available at the beginning of the DataGrid project, and refer to (García and Peso, 2002) for a more detailed analysis of the tools that have been specifically designed for the installation and management of Linux clusters.

At the beginning of the DataGrid project, none of the tools appeared to fulfill all the requirements needed for managing a grid computing fabric. The tool more suited to the DataGrid project requirements were LCFG (Anderson and Scobie, 2002).

LCFG (*Local Configurator*) is a system for the installation and configuration of nodes, originally developed at the University of Edinburgh. LCFG has been designed to automatically install and manage a network of computers with very different configurations, which may evolve rapidly over time. LCFG is based on the use of a central repository where all the node configuration parameters are stored in an abstract way, and organized into categories. A simple mechanism for file inclusion provides inheritance, and a modified form of inheritance called mutation. This central repository for configuration, and the abstract representation of the configuration parameters, are the key elements of LCFG. Moreover, there exists a set of scripts in the client nodes that read the abstract configuration parameters, and they generate the traditional configuration files, so as to directly manage the services of the node (Barroso et al, 2001).

LCFG was chosen by WP4 as a interim solution for the installation and configuration of DataGrid software, and as a prototype for testing the architectural design principles of quattor.

## 2. Architecture and Design

The quattor information model is based on the distinction between the *desired configuration state* and the *actual configuration state* of the fabric nodes. The *configuration management* modules are responsible for describing the desired configuration of all the nodes, and the *installation management* modules are responsible for changing the actual configuration of nodes to the desired state.

### 2.1. Design Principles

The most relevant design principles followed by the quattor developer team are:

- *non-intrusive system*: quattor does not replace any of the native installation and configuration tools that already exist for Linux, instead quattor adds extra functionality on top of them,
- *highly modular*: quattor provides independent modules that can work together as a complete solution, but that can also be taken independently and cleanly interfaced to other fabric management tools,
- *use of well known standards*: the HTTP protocol is used to transfer configuration information and software packages, XML to describe configuration information, SOAP for remote management of quattor services,
- *use of third party tools*: quattor reuses many installation and configuration tools (KickStart, PXElinux, DHCP, etc.), in this way the system is easier to maintain,
- *distributed approach*: configuration information is cached locally on clients, and operations are performed locally whenever possible, in this way the scalability of the system is guaranteed, and
- *provides a set of base libraries*: these libraries let users access the modules via control functions.

### 2.2. Configuration Management

This section describes the modules provided by quattor for the management of the desired configuration state of client nodes. The relationship among these modules can be seen in Figure 1.

#### 2.2.1. *Configuration Database*

The desired state of client nodes is registered in a central fabric-wide database called the *Configuration Database* (CDB). This database stores configuration information using two different formats. One format is a high level description of configuration parameters intended for user-level processing. This high level format is expressed in a specially designed configuration language called Pan (see below for more information about Pan). The other format is a low level description

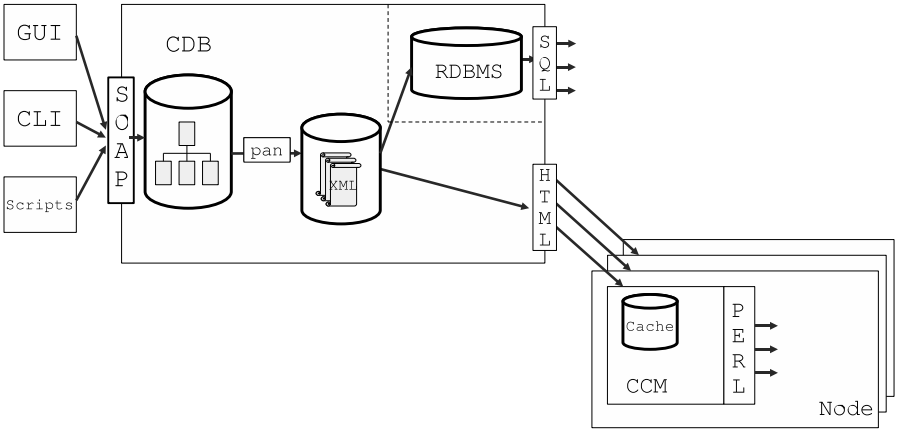


*Figure 1.* Configuration Management

expressed in XML, called the *node profile*, and intended for machine processing. The low level XML files are derived from the high level Pan descriptions using the Pan language compiler.

System administrators edit the high level description files, called *configuration templates*, either through a command line based interface, or using a graphical user interface. These templates describe the configuration parameters of all the elements that belong to the fabric. Then, for each client node, one XML file is generated that contains all the information needed to install and configure the node.

The CDB works in a transactional way. It performs validation of the configuration information. Once the validation and compilation process is accomplished successfully, the changes introduced by the user are stored in the CDB and are visible to its clients. The CDB is also able to manage concurrent access by different system administrators. Additionally, it provides mechanisms for versioning (based on CVS) and it maintains the history of all changes to the configuration information.

The database itself includes a scalable distribution mechanism for the XML files based on HTTP, and the possibility of adding any number of back-ends or *Server Modules* to support various query patterns over the information stored in the CDB (currently SQL queries).

### 2.2.2. *Pan Language*

The configuration information is structured in a tree format, and expressed with the High Level Description Language called Pan (Cons and Poznański, 2002). Pan mainly consists of statements to set some value to a configuration parameter identified by its path in the configuration information tree.

Pan features include other statements like `include` (very similar to C++ `#include` directive) or `delete` that removes a part of the configuration information tree.

The grouping of statements into templates allows the sharing of common information and provides a simple inheritance mechanism

Pan contains a very flexible typing mechanism. It has several built-in types (such as "boolean", "string", "long" and "double") and allows compound types to be built on top of these. Once the type of an element is known, the compiler makes sure that only values of the right type are assigned to it.

To have even greater control over the information generated by the compiler, it is possible to attach validation code to a type or to a configuration path.

The validation code is represented in a simple but powerful data manipulation language which is a subset of Pan and syntactically similar to C and Perl.

#### 2.2.3. *Configuration Cache Manager*

The *Configuration Cache Manager* (CCM) runs on every node and caches the XML machine configuration to support disconnected operations. Access to the configuration information is provided through a Node View Access API that hides details such as the XML schema used.

The CCM may poll the CDB for the configuration information. It also receives UDP notification sent by the CDB if the machine's configuration is changed. Once validated, configurations are propagated to CCM agents running on the nodes and cached locally.

The CCM is also able to store multiple profiles and provides a locking mechanism to forbid concurrent access.

### 2.3. Installation Management

This section describes how quattor deploys the desired configuration state on the fabric's nodes. The relationship among the installation modules can be seen on Figure 2

#### 2.3.1. *Node Configuration Manager*

The *Node Configuration Management* (NCM) module provides a framework for adapting the actual configuration of a node to its desired configuration, as it is described in the node's profile inside CDB.

Plug-in software modules called *configuration components* are responsible for the configuration of local services (e.g. network, users and groups, NFS). These components read configuration information stored

in the CDB via the CCM, and create, update or delete local service configuration files in order to match the CDB configuration description. Components register which configuration entries or subtrees they are interested in, and are notified if a change occurs.

Each component contains the knowledge for the translation of the CDB configuration information into each local service's specific configuration file syntax. A component may also need to notify a service about a configuration change (e.g. by running a `restart` or `reload` method in a SysV init script).

The NCM module is composed of the following programs and libraries:

- A *Configuration Dispatch Daemon* (`cdispd`) that monitors the node configuration profile by polling the CCM. In case of changes in the configuration profile, `cdispd` will invoke the affected configuration components.

- A *Node Configuration Deployer* (`ncd`) is the framework and front-end for executing the configuration components. The `ncd` program can be executed manually, at regular intervals of time via cron utility, or when there are changes in the configuration via the `cdispd`. It takes care of configuration locking and inter-component dependency ordering prior to executing components sequentially.

- A set of *Component Support Libraries.* Libraries for recurring system management tasks (system information, interfaces to system services, file editing), and a *Common Application Framework* with utilities for log file handling, command line processing, and so on.

#### 2.3.2. *Software Package Management*

The *Software Package Management and Distribution* (SPM) module is responsible for managing and storing software packages, and for distributing and installing these packages on client nodes.

The SPM module is composed of:

- A *Software Repository* (SWRep) module that allows site administrators and package maintainers to store and manage software packages (for example Linux RPM or Solaris PKG packages) subject to authentication and authorization using access control lists. The packages themselves are accessible to clients via standard protocols including HTTP, FTP, or via a shared file system. It is possible to have multiple (replicated or independent) Software Repository instances for a given fabric, allowing private per-department

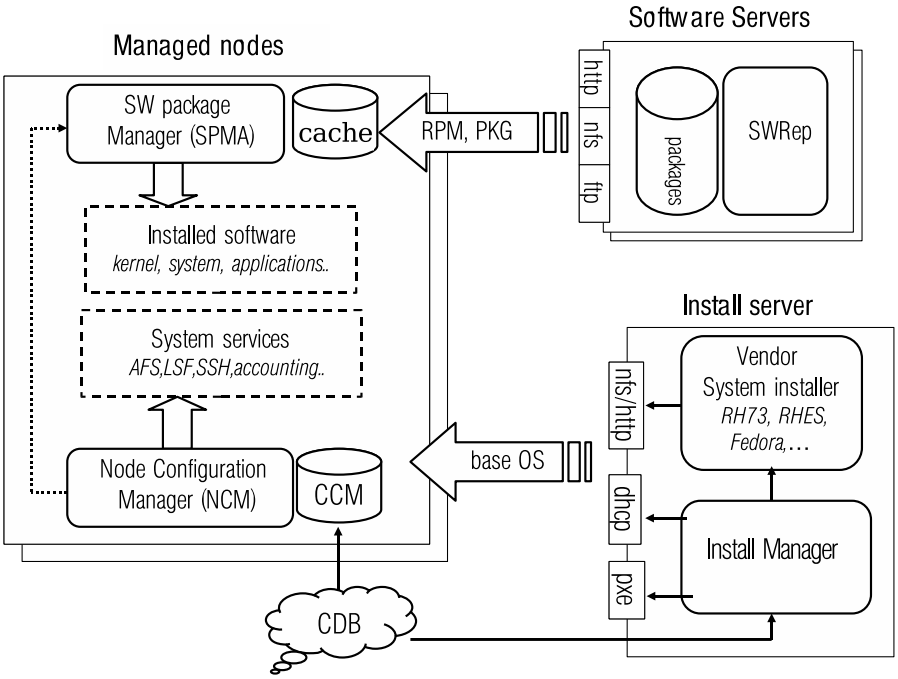


*Figure 2.* Configuration Deployment

repositories. The replication of repositories can be done with standard tools like `rsync`.

- A Software Package Manager Agent (SPMA) that runs on the target nodes. It reads a local configuration file containing the list of desired packages, compares this list with the currently installed packages, computes the necessary installations, deinstallallation or upgrades operations, and invokes the system packager (e.g. rpm[2] on Linux or `pkgadd/pkgrm` on Solaris) with the correct transaction set. The software packages can be cached locally if necessary before calling the node's system packager.

- An NCM configuration component: The information on which packages are to be deployed on the nodes (desired or target configuration), and which packages are available on every repository can be kept in the CDB. The SPM component fits into the NCM framework described above. It retrieves the list of packages to be installed for the current node from the CDB via the CCM, creates with this information a local configuration file for the SPMA, and launches the SPMA.

Typically, the SPMA is used for managing all packages on a node. This is useful for nodes which are under full control of the fabric management system. However, for add-on installations or desktop systems, the SPMA can be run in a "light" mode, taking care of a subset of packages only, according to configurable policies.

For performance and scalability issues, the SPMA can use a local cache where packages can be stored in advance of install/upgrade operations. This way, peak loads on software repository servers can be avoided during upgrades of large farms, while keeping consistency across the upgraded nodes. The default transport protocol is set to HTTP for its scalability and low overhead.

#### 2.3.3. *Automated Installation Infrastructure*

The *Automated Installation Infrastructure* (AII) module provides tools for the initial installation of the operating system and the quattor software on client nodes. AII uses standard vendor installation tools and network bootstrap protocols, providing a common front end for configuring and managing these tools. AII manages the configuration of network related information, like the DHCP tables and the network bootstrap protocols (e.g. PXE for Linux and OpenBoot for Solaris), and the node specific installation setup rules (Kickstart for Red Hat Linux and JumpStart for Solaris). AII obtains the configuration information about the nodes to bootstrap from the CDB (via CCM), but it can be used with other site specific network databases as well. The software packages used to install the clients can be downloaded from the SWRep repository.

## 3. Results

At the time of writing this paper, most of quattor's modules are in a production state and being used to manage a very large computing fabric at CERN. CERN is progressively replacing its legacy management tools (SUE (Tobbicke, 1995) and ASIS (Defert, 1997)) by quattor. The non-production quality quattor modules, essentially AII and NCM's `cdispd`, are still undergoing heavy testing in a fabric testbed at the UAM University of Madrid.

### 3.1. Usage Statistics

At the CERN computing centre, quattor is responsible for the installation and management of more than 2000 client nodes. These nodes belong to 15 different clusters, being used for different purposes (interactive clusters, batch processing, etc.). The client nodes managed by

quattor have many different functionalities, such as computing nodes, disk and tape servers, database and web servers, and others. Moreover, the managed nodes have heterogeneous hardware, for example, different memory sizes and hard disk sizes. Quattor is used to manage three different versions of Linux: Red Hat 7.3, Red Hat Enterprise Server 2.1 and Red Hat Enterprise Server 3.0.

Quattor is also responsible for the management of more than 400 grid computing nodes. These nodes are installed under LHC[3] Computing Grid software (LCG). Among these nodes, 250 are being used for the LHC CMS detector data processing challenge. The remaining nodes are used for LCG (version 2) grid production.

The Software Repository used at CERN contains more than 8000 software packages, divided into 4 platforms, with 7 areas in each one. This repository is managed by 9 different system administrators.

The CERN Central Configuration Database stores 260 general purpose configuration templates. These templates describe the roles of all the types of node used in the 15 clusters (an example of role could be an LCG's *Worker Node*). Beside the general purpose templates, for each client node there are 2 additional templates describing its specific configuration parameters, such as the node's IP address. In total, CERN's CDB manages more than 3800 configuration templates.

### 3.2. Software Package Management

Quattor's SWRep repository and NCM/SPMA modules are used at CERN for the complete automated management of all software packages, operating system packages, grid middleware and application software. Quattor is used to install software packages updates that fix security loopholes. On average, SPMA is run every two days to keep the fabric updated to the latest software release and bug fixes.

Quattor can be used to manage the installation, upgrading or downgrading of a large number of software packages in just one single transaction. For example, an upgrade of the KDE desktop environment equates to approximately 400 MB of software packages per node, and these packages must be upgraded in more than 70 nodes of the interactive cluster, or the OpenSSH software that must be upgraded on more than 2100 nodes.

A critical software element managed by quattor is the upgrading of Linux kernels. SPMA is capable of installing and managing multiple versions of the same package. In this case, it can install multiple kernel versions on the same machine. With quattor we can distinguish between the time of kernel installation, and the time of kernel activation. With NCM and the kernel loader configuration component, we can configure

which kernel version we want to use each time the machine reboots in an easy way: modify a single configuration template, and all the machines will be automatically updated.

One remarkable example of the utility of quattor was the LSF batch system software upgrade. The challenge was to upgrade from LSF 4.2 to LSF 5.1 on more than 1000 cluster nodes in the shortest possible time, and without interrupting production service. Using quattor, this task was performed in less than 15 minutes, and without service interruption. The previous major version upgrading (from LSF 3 to LSF 4), performed on 600 nodes took two full time system managers during three weeks to complete. A key element of this success was quattor's package pre-load caching facility, and the central storage of configuration information in the CDB.

The quattor software management tool SPMA has the advantages over other tools like `apt-get` that is not only can install and upgrade software packages, SPMA can be used to downgrade to any older version of the software. This facility is very useful in case of mistakes to rollback to a previous well-know state. The SPMA pre-load cache facility seems to be very useful to reduce the time of upgrading critical software components.

### 3.3. Proxy Servers

One issue in the early use of quattor has been how to provide reliable, redundant and load balanced access to quattor services (such as AII, SWRep and CDB profiles).

The previously used deployment architecture at CERN can be seen in Figure 3. The following problems were found using this previous architecture:

- scalability: this is limited, due to bottlenecks in network and switches, for example,
- efficiency: between servers, the entire disk content must be replicated (with implications for disk size and access speed), while between servers and clients, much bandwidth is wasted, owing to the multiplication of identical transfers, and
- reliability: given a low ratio of servers to clients (e.g. a handful of servers coping with 1000s of client nodes), then load balancing must react very quickly and intelligently in the case of server unavailability.

Quattor modules are by design reverse proxy compatible. For example, SPMA has support for proxy handling. The current architecture

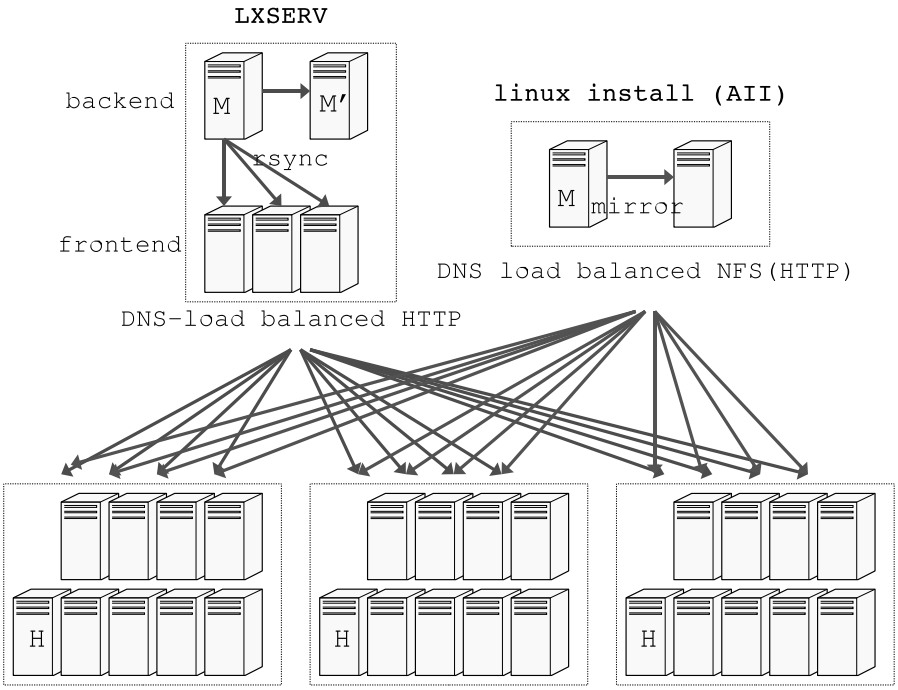


*Figure 3.* Old Deployment Architecture

deployed at CERN is based on a two-level proxy server hierarchy, using Apache's `mod_proxy` module. This new architecture can be seen on Figure 4.

The early experience with Apache show that is a reliable web server, with a clear and flexible configuration, and with no functionality duplication (it can be used as proxy and non-proxy). All problem related to scalability, efficiency and reliability has been solved.

## 4. Conclusion

Fabric management components are not grid components themselves, but they are essential to have a working grid. Our experience gathering requirements, developing and deploying the quattor tool suite in the DataGrid testbed and the CERN Computing Centre shows that there is a real need to be able to install, configure and manage grid computing clusters correctly (to avoid configuration errors that may affect not only the site but the whole grid response), automatically (to reduce the work load of system administrators), and supporting adaptability (properly managing site and resources reconfigurations).

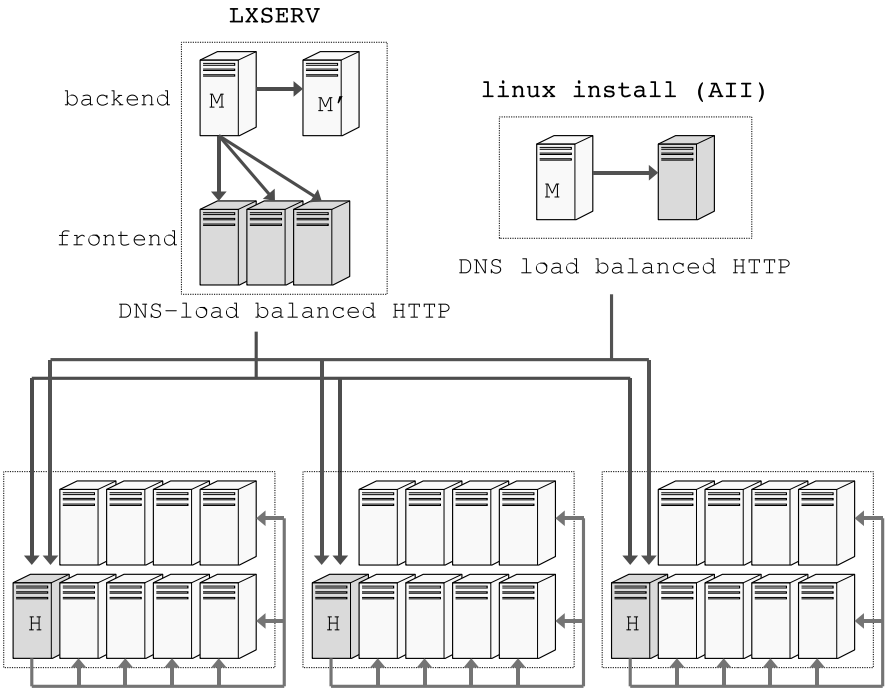


*Figure 4.* Proxy based Architecture

While the autonomy model for sites participating both in testbeds and production grid computing fabrics is the preferred solution, the proper integration of these sites into grid of ever increasing size and complexity necessarily requires fully automated fabric management solutions and precisely defined policies which must be binding for all participating sites. This is critical for the manageability, stability and security of grid computing fabric, and will also help in keeping the running cost within a reasonable range.

During the time life-spam of the DataGrid project, a prototype solution for fabric management (LCFG) has been used on site test beds. The experience and early feedback from users and site administrator with this prototype, was a very valuable input to quattor's developers team, who finally produced a solution that better fits the requirements of grid fabric environments.

The experience gathered shows that it results very expensive to change the current fabric management tools used at grid sites. The fabric management tools are at the bottom of everything, usually, managing all the components, and to change the current tool by a new one implies to learn the new tool, that has new procedures, new

methods, and so on. In case of the DataGrid sites, these were testbed where tools and procedures could be imposed. But this is not the case for real production grid fabrics.

Necessarily, there must be a time where the new tools has to coexist with the legacy tools, services and procedures. Quattor has a modular design and with very clean interfaces so it can be incrementally replaced. CERN is replacing its legacy management tools by quattor.

The scalability of the quattor tool suite has been tested at CERN/IT computing centre by managing 15 clusters, with more than 2000 machines. The machines managed by quattor have different roles (CPU worker nodes, tape and disk servers, infrastructure servers), and many different hardware. The use of proxy servers together with quattor seems to guarantee the scalability of the proposed solution to very large grid fabric environments.

### 4.1. Current State and Future Work

Most of the quattor subsystems are already available. The primary platform is Red Hat Linux, but platform independence is a design principle (ports and plug-ins for Solaris are progressing). The next steps for quattor tool suite are the following:

- Development of a full set of grid configuration components: actually quattor can be used to install the Worker Nodes of DataGrid and LCG2 (globus tool suite, replica manager, etc). A full set of configuration components for a grid is underway. These grid configuration components are being written using the experience gathered with LCFG objects.
- Move all modules to production quality: test the Automated Installation Infrastructure and the Configuration Dispatch Daemon in a real, large, production fabric environment.
- Test further the scalability: CERN plans to use quattor to manage more than 6000 nodes in 2006, for the LHC project.
- LCG deployment: quattor is being evaluated by LCG Tier_1 centres for the future deployment of LCG grid software.

## Notes

[1] quattor stands for QUattor is an Administration ToolkiT for Optimizing Resources.

[2] Since `rpm` on Linux does not accept multiple simultaneous operation types, we developed a new front end called `rpmt` (for transactional `rpm`) capable to handle multiple operations on multiple packages in a single transaction.

[3] CERN's new Large Hadron Collider.

## References

Bärring, Olof et. al., "Towards automation of computing fabrics using tools from the fabric management workpackage of the EU DataGrid project", *CHEP Conference Proceedings*, 2003.

Cancio Meliá, Germán (Ed.), "WP4 Architectural Design and Evaluation Criteria", *DataGrid Technical Report DataGrid-04-D4.2-0119-2-1, Nov. 2001.*

Barroso, Maite, "WP4 Report on Current Technology", *DataGrid Technical Report DataGrid-04-TED-0101-3_0*, Apr. 2001

Garca Leiva, Rafael y Peso, Jose del, "Open Source Solutions for Installation and Management of PC Clusters under Linux for ATLAS", *ATLAS Notes ATL-SOFT-2003-001*, Oct. 2002.

Anderson, Paul and Scobie, Alastair, "LCFG: The Next Generation", *Proceedings of UKUUG Winter Conference, 2002.*

Cons, Lionel and Poznański, Piotr, "Pan: A High-Level Configuration Language", *LISA Conference Proceedings*, 2002.

Tobbicke, R. (ed.) SUE Definition Document, May 1995

Defert, Ph. et al. ASIS, manage and distribute application software in the HEP community, *CHEP Conference Proceedings*, 1997.