ADVANCES IN FABRIC MANAGEMENT BY THE CERN-BARC COLLABORATION

Murthy Chandragiri, BARC, Mumbai, India
Dinesh Sarode, BARC, Mumbai, India
William Tomlin, CERN, Geneva, Switzerland

Abstract

The collaboration between BARC and CERN is driving a series of enhancements to ELFms[1], the fabric management tool-suite developed with support from the HEP community under CERN's coordination. ELFms components are used in production at CERN and a large number of other HEP sites for automatically installing, configuring and monitoring hundreds of clusters comprising of thousands of nodes. Developers at BARC and CERN are working together to improve security, functionality and scalability in the light of feedback from site administrators. In a distributed Grid computing environment with thousands of users accessing thousands of nodes, reliable status and exception information is critical at each site and across the grid. It is therefore important to ensure the integrity, authenticity and privacy of information collected by the fabric monitoring system. A new layer has been added to Lemon, the ELFms monitoring system, to enable the secure transport of monitoring data between monitoring agents and servers by using a modular plug-in architecture that supports RSA/DSA keys and X509 certificates. In addition, the flexibility and robustness of Lemon has been further enhanced by the introduction of a modular configuration structure, the integration of exceptions with the alarm system and the development of fault tolerant components that enable automatic recovery from exceptions. To address operational scalability issues, CCTracker, a web-based visualisation tool, is being developed. It provides both physical and logical views of a large Computer Centre and enables authorised users to locate objects and perform high-level operations across sets of objects. Operations staff will be able to view and plan elements of the physical infrastructure and initiate hardware management workflows such as mass machine migrations or installations. Service Managers will be able to easily manipulate clusters or sets of nodes, modifying settings, rolling out software-updates and initiating high-level state changes.

THE PROBLEM

In a complex environment such as CERN's Tier 0/1 Computer Centre, it is necessary to rapidly evolve processes and tools to prepare for LHC-scale operations. In particular:

- The configuration and monitoring of up to 10,000 heterogeneous nodes with diverse functionality
- Mass installations, moves and retirements
- Daily hardware failures

The response to this need is being met by the 'Extremely Large Fabric Management System' (ELFms[1]).

THE EVOLUTION OF LEMON

Authentication

LEMON[2], the 'LHC-Era Monitoring' component of ELFms, is being used at CERN to monitor approximately 80 metrics across 100 clusters comprising of 2,500 nodes. These metrics contain the performance and exception information necessary for effective cluster management. It is a strong requirement that this information is authentic and accurate. The repository server must be able correctly verify that the samples are from the stated agent and that data integrity is preserved. To achieve this, a security layer has been added to LEMON which enables the signing of information by the agent and corresponding signature verification by the server. The server drops samples whose signature has not been verified. This mechanism establishes the trust relationship between agents and the repository server, thus freeing the server from processing samples above the transport layer and avoiding wasted CPU and storage capacity.

The Process – Agent View

A LEMON agent may send monitoring samples to a number of servers on transport channels using UDP or TCP. Each transport channel has an associated configuration. The security layer allows selectively enabling or disabling authentication for each channel. Processed samples from LEMON sensors are arranged in transport buffers before being written to send sockets. At this stage, if authentication is enabled, a digital signature is created and pre-pended to the data in the message buffers along with the agent's ID and digest type. The whole message is then sent on the transport socket.

The Process – Server View

The LEMON server receives messages from agents as either TCP or UDP packets and subjects them to parsing in order to store the samples in the repository. The security layer adds a digital signature verification phase prior to parsing the message. Upon receiving the message, the server identifies whether it is signed or

unsigned. In the case of a signed message, the data is passed into the verification phase. In this phase, the digital signature is verified using the public key of the agent and the digest-type information contained in the message. If the verification fails, the samples are dropped, otherwise they are passed on to the next stage of processing.

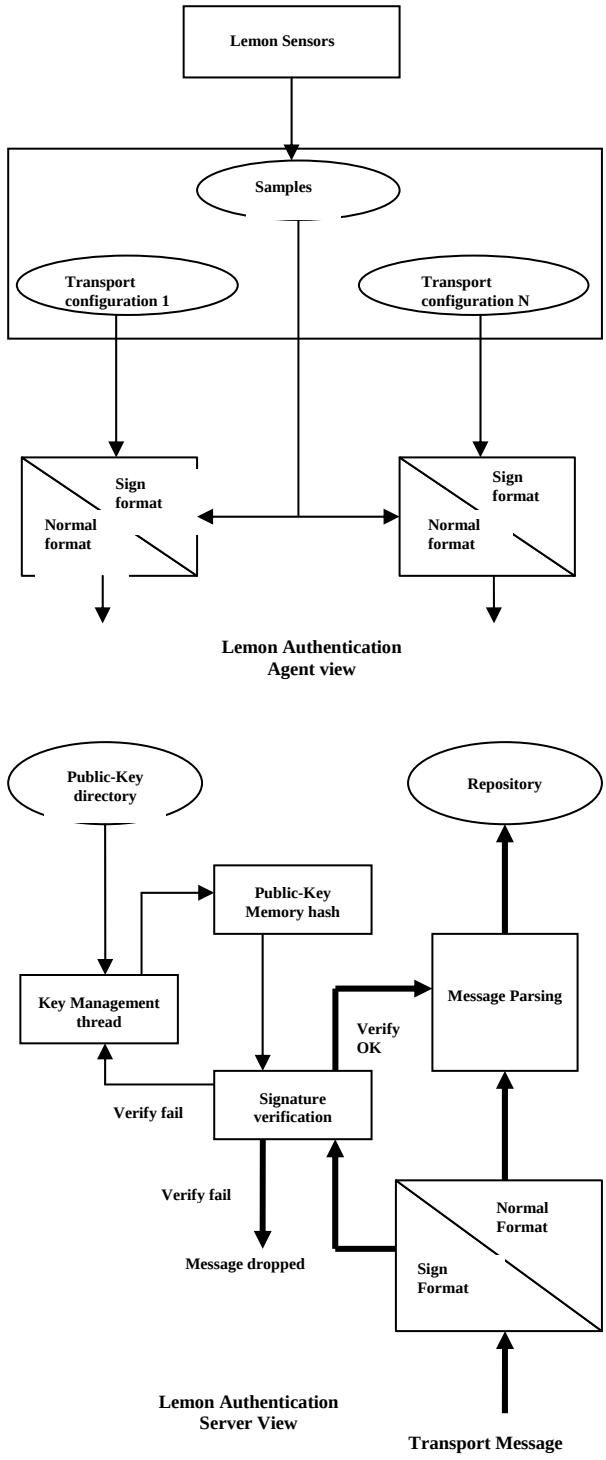

Figure 1: LEMON Authentication

The server also employs a separate thread for asynchronous management of public key information from agents. This thread starts during server initialization and waits for key update requests from the server core. On any verification failure, the server requests from this thread a public key information update for the agent in question. The thread then re-reads the public key directory for key files and dynamically updates the key information in memory.

Correlations and Automatic Recovery

In environments with thousands of nodes, the ability to raise exceptions based on simple or correlated metrics and to automatically recover from these exceptions where possible is essential to providing a high quality of service and to liberate service managers from repetitive tasks, such as cleaning up temporary space or restarting daemons.

An early EDG fault-tolerance prototype has been re-engineered and now provides two independent modules for global and local scope action.

A light-weight module running on every monitored node can compare cached metric samples to absolute or relative reference values and can trigger recovery actions or raise exceptions which lead to the triggering of an operator alarm. Some examples of the exceptions configured at CERN are:

- High CPU load
- File system errors reported in /var/log/messages
- syslogd not running in the system
- Swap file system occupancy > 98%

The correlation engine has been redeveloped to provide recovery actions on complex logical conditions. For example, if metric x has some value and metric y another value, then this can give rise to an exception which will invoke an actuator to repair the problem. If the recovery action fails a configurable number of times, then an alarm is raised to the operator. To date, some 80 automatic recovery actions have been put into production across 2,500 nodes.

Lemon has a new sensor, called ‘alarm.exception’, that supports this evaluation of logical and arithmetic expressions across multiple metrics and raised alarms.

An actuator process can be defined to take corrective action to resolve a specific problem. After the actuator executes, the exception condition is checked again asynchronously to check whether it still persists.

The sensor allows the configuration of the maximum number of times an actuator can run consecutively within a specified time window, the maximum amount of time an actuator can run before being killed because of a timeout and the amount of time to wait to resample the exception metric itself after the execution of the actuator. For example, the metric syslogd_wrong raises an alarm whenever there is no syslogd daemon running on the node.

An actuator module is defined for this exception, which restarts the daemon. This actuator is configured to run as many as 3 times, each time restarting the daemon. The failure to find the daemon again after this many invocations will finally raise an operator alarm to indicate that follow-up is needed.

Configuration Changes

In the past, a large configuration file housed all settings for sensors and transports on each subscribing node. Now, a modular scheme is in place where each sensor and set of server connection parameters has a separate configuration file in the style of xinit.d. This means that new sensors can easily be dynamically added or reconfigured without affecting others.

CCTRACKER

Overview

A second major area of development in the collaboration on fabric management has been the development of a web-based tool, CCTracker, to visualise and simplify the management of a large Computer Centre, such as those at CERN and BARC.

The intention is to provide a portable, highly functional client that may be used to manage a complex environment comprising of multiple separate rooms with diverse object types.

Requirements

The Use Case analysis phase of the project yielded the following requirements:

- Display a physical view of each room that comprises the Computer Centre, including racks and tape equipment
- Display a logical, hierarchical view of objects and their properties
- Display a service-level overview with links to service specific information
- Perform client-side caching of information to allow location of equipment during server failures

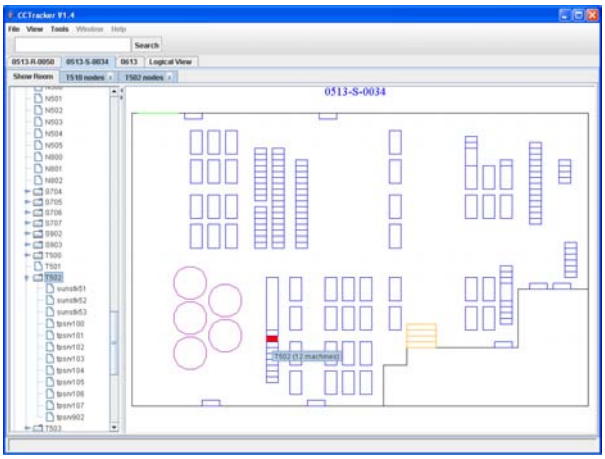

Figure 2: CCTracker Physical View

- Enable the selection of arbitrary sets of objects to perform high-level operations
- Initiate service management Use Cases such as moving nodes automatically in and out of production, kernel upgrades, operating system upgrades and cluster re-assignments to sets of nodes
- Initiate system administration Use Cases such as IP renumbering, reinstallations, shutdowns and restarts to sets of nodes, ssh'ing to the node or via serial console from the head node as required
- Initiate operational Use Cases such as drawing planned racks and physically moving sets of machines
- Design for portability, configurability and extensibility

Design

The CCTracker client is a Java Web Start application that consumes XML over HTTP in order to present physical and logical information to the user. Web services are leveraged in order to perform high-level update actions across sets of objects.

The design makes a clear separation between generic and site-specific components and is highly configurable (see Figure 3).

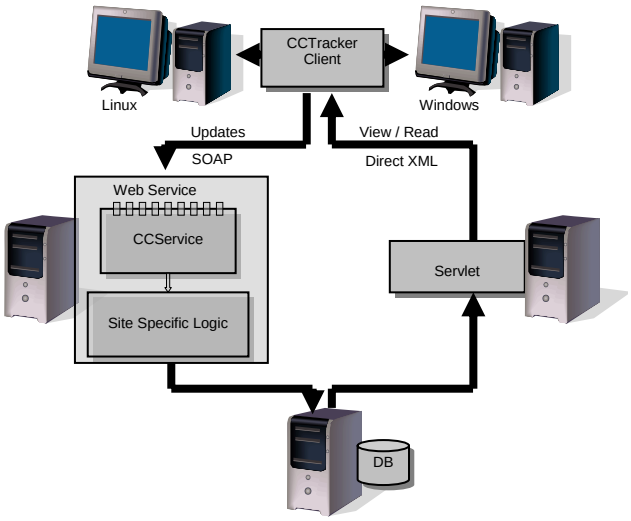

Figure 3: CCTracker Design

The CCTracker client and the CCService web service are site-independent components. In order to provide a display-only tool, a site is only required to implement a server-side component that provides compliant XML via the web.

To invoke actions from the client, the CCService component, an AXIS SOAP server, must be deployed and

each site must implement the underlying business logic to perform the required operations, including authentication.

Configuration

High configurability is achieved through the definition of a URL in the application's JNLP (Java Network Launching Protocol) file which points to a 'bootstrap' XML available via the web.

The bootstrap XML defines all site specific and configurable elements needed by the client, such as the URL pointing to the room layout XML.

For example, at BARC, the XML is dynamically generated by servlets accessing a PostgreSQL database. At CERN, the XML is generated dynamically by Oracle PL/SQL stored procedures web-exposed via the Oracle Application Server. As long as the XML schema is respected, CCTracker will present the content it receives, regardless of how or where that content is generated.

Deployment

CCTracker is currently in production at CERN and displays logical and physical views, search capabilities, links to external monitoring pages, an administration tool and a refresh feature. It is planned to be in production at BARC by the end of Q1, 2006 for visualisation.

Testing is currently underway for client-side caching and adding physical equipment such as racks and modifying object properties.

The plan is that, by the end of 2006, all of the update functionality currently specified will be fully tested and available in production. In addition, documentation will be available that may encourage other sites to adopt the software.

CONCLUSION

To deliver high availability grid services on a massive scale, it is essential that highly automated, self-repairing local fabric management components are in place.

Discovering problems as early as possible, or even pre-emptively, and then resolving them automatically wherever possible, can greatly improve the overall quality of service and is a highly cost-effective approach. A problem discovered at the grid service level is significantly harder to diagnose and resolve than one caught early on and dealt with at the local level, preferably without human intervention.

The exchange of ideas, experiences and solutions between BARC and CERN has proved very beneficial to both parties, with BARC planning to adopt several ELFms components for their operations, including CCTracker and LEMON, and CERN benefiting from additional skilled resources.

Aside from the significant value of the interchange of knowledge, the collaboration has resulted in a more portable, robust, scalable and functional fabric management tool-suite. A solid basis for LHC-scale operations is in place.

ACKNOWLEDGEMENTS

The authors would like to thank the other members of the CERN-BARC collaboration on fabric management: German Cancio-Melia (CERN), P.S. Dhekne (BARC), Ramgopal Mundada (BARC), Sharma Rohitashva (BARC), Sonika Sachdeva (BARC), Miroslav Siket (CERN), Dennis Waldron (CERN).

REFERENCES

[1] http://www.cern.ch/elfms
[2] http://www.cern.ch/lemon