A self-configuring control system for storage and computing departments at INFN-CNAF Tier1

Daniele Gregori[1], **Alessandro Cavalli**[1], **Matteo Favaro**[1], **Pier Paolo Ricci**[1], **Michele Pezzi**[1], **Andrea Prosperini**[1], **Vladimir Sapunenko**[1]

[1] INFN-CNAF, Viale Berti-Pichat 6/2, 40127 Bologna, Italy

E-mail: `daniele.gregori@bo.infn.it`

Abstract. The storage and farming departments at the INFN-CNAF Tier1[1] manage approximately thousands of computing nodes and several hundreds of servers that provides access to the disk and tape storage. In particular, the storage server machines should provide the following services: an efficient access to about 15 petabytes of disk space with different cluster of GPFS file system, the data transfers between LHC Tiers sites (Tier0, Tier1 and Tier2) via GridFTP cluster and Xrootd protocol and finally the writing and reading data operations on magnetic tape backend. One of the most important and essential point in order to get a reliable service is a control system that can warn if problems arise and which is able to perform automatic recovery operations in case of service interruptions or major failures. Moreover, during daily operations the configurations can change, i.e. if the GPFS cluster nodes roles can be modified and therefore the obsolete nodes must be removed from the control system production, and the new servers should be added to the ones that are already present. The manual management of all these changes is an operation that can be somewhat difficult in case of several changes, it can also take a long time and is easily subject to human error or misconfiguration. For these reasons we have developed a control system with the feature of self-configure itself if any change occurs. Currently, this system has been in production for about a year at the INFN-CNAF Tier1 with good results and hardly any major drawback. There are three major key points in this system. The first is a software configurator service (e.g. Quattor or Puppet) for the servers machines that we want to monitor with the control system; this service must ensure the presence of appropriate sensors and custom scripts on the nodes to check and should be able to install and update software packages on them. The second key element is a database containing information, according to a suitable format, on all the machines in production and able to provide for each of them the principal information such as the type of hardware, the network switch to which the machine is connected, if the machine is real (physical) or virtual, the possible hypervisor to which it belongs and so on. The last key point is a control system software (in our implementation we choose the Nagios software), capable of assessing the status of the servers and services, and that can attempt to restore the working state, restart or inhibit software services and send suitable alarm messages to the site administrators. The integration of these three elements was made by appropriate scripts and custom implementation that allow the self-configuration of the system according to a decisional logic and the whole combination of all the above-mentioned components will be deeply discussed in this paper.

1. Introduction

The departments Storage and Farming at the INFN-CNAF Tier1 administer around a thousand computing nodes and some hundreds of servers. A control system based on the states of services and hosts, able to send email alerts and automatic recovery actions in case of error is an essential tool. Given the large number of machines to be controlled it becomes very difficult to keep aligned the control system with the machines actually in production. For this reason, we decided to realize a control mechanism able of self configuration according to the machines really in production and to their role in terms of the type of server (e.g. the IBM General Paralle File System GPFS, GridFTP, Worker Node, User Interface). For this purpose we have used a set of standard open source tools: Quattor[2], Nagios[3] and a Postgres database containing information on servers in the computing center named Docet[4]. All these tools have been integrated together by means of a series of Bash scripts that have the purpose of reading the information on the machines in production and creating the updated Nagios configuration. In the following sections, the tools used and the operating logic of integration will be described.

2. Adopted Tools

2.1. Quattor

Quattor is a generic open-source tool-kit used to install, configure, and manage computers; it ensures that all machines that need to be monitored by Nagios are configured to perform the required checks.

2.2. Nagios

Nagios is an open source computer system monitoring, network monitoring and infrastructure monitoring software application. Nagios offers monitoring and alerting services for servers, switches, applications, and services. It alerts the users when things go wrong and alerts them a second time when the problem has been resolved.

2.3. Docet

Docet is a PostgreSQL database (DB) that contains all hardware information on all the machines in production at the Tier1. The information can be extracted from the DB and used to define which machines are actually in production and therefore which need to be controlled via Nagios.

3. Project Constraints

At the time of system design has taken into account what are the specific requirements of each department. In particular for farming the large number of computing nodes ~ 1000 must be taken into account. For the storage unit, the configuration of the GPFS cluster is an element that must be taken into consideration in order to control the specific GPFS server nodes as quorum.

- The system must be able to control thousands of Worker Node and in the case of problems must be able to perform automatic recovery actions.
- The system must be able to automatically learn which machines (worker node or server) should be monitored and how the GPFS clusters are configured.
- The system must be quickly restored in case of failure of the control server (disaster recovery).

The Worker Nodes of the farming department must check a list of running daemons and mounted filesystems. In case of problems the control system has groped a restart of the services or mount any filesystem that is not mounted. Given the high number of nodes and controls to be done Nagios is not able to perform active controls (synchronous) via plugin NRPE[5]. To fulfill this purpose we employ passive controls (asynchronous) via plugin NSCA[6]. On the Worker Node a series of sensors to control the demons and the file system are performed within an hour without setting the exact time using a random delay. They send their results to the Nagios server on a named pipe, the Nagios server in this way reads the value of the service

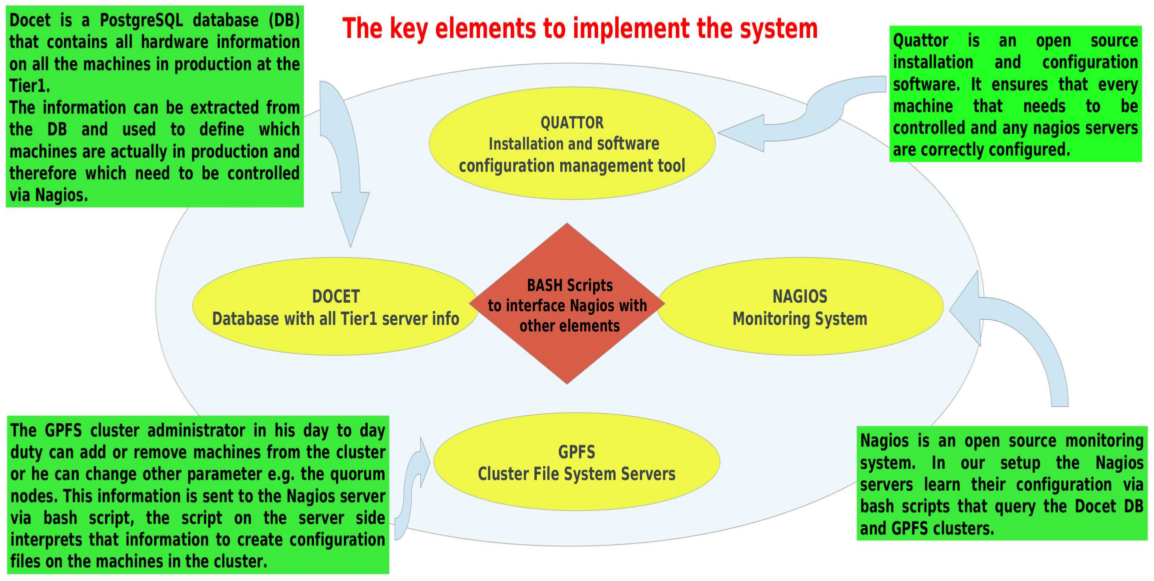

The key elements to implement the system
Docet is a PostgreSQL database (DB) that contains all hardware information on all the machines in production at the Tier1.
The information can be extracted from the DB and used to define which machines are actually in production and therefore which need to be controlled via Nagios.
Quattor is an open source installation and configuration software. It ensures that every machine that needs to be controlled and any nagios servers are correctly configured.
QUATTOR
Installation and software configuration management tool
DOCET
Database with all Tier1 server info
BASH Scripts to interface Nagios with other elements
NAGIOS
Monitoring System
GPFS
Cluster File System Servers
The GPFS cluster administrator in his day to day duty can add or remove machines from the cluster or he can change other parameter e.g. the quorum nodes. This information is sent to the Nagios server via bash script, the script on the server side interprets that information to create configuration files on the machines in the cluster.
Nagios is an open source monitoring system. In our setup the Nagios servers learn their configuration via bash scripts that query the Docet DB and GPFS clusters.

Figure 1. Overview of the key elements of the control system.

status checked. If there is no state sent for service for more than an hour Nagios status changes to “Unknown” and a mail is sent to the administrators of that service. If from the Worker Node sent a status of “Error” then Nagios will react immediately, proactively via NRPE plugin, requesting an attempt to restart the service and then send the new state. For this type of machine Nagios learns which should check through query to the Docet DB. The request besides the list of worker node in production provides the hostname, the types of hardware, whether they are physical or virtual machines, and the network switch to which they are connected. Based on this information, the Bash scripts construct the Nagios configuration and make an initial syntax checking and duplication of IP addresses. At this point the reload of the configuration is left to the Nagios administrator.

The requirements for storage department are different. There are numerous GPFS cluster that must be controlled, the server within the cluster can be added or removed, or may change their role as quorum manager. All this information is not contained in the Docet DB, so in order to learn the cluster configuration one of the cluster servers must communicate with Nagios. The script `/opt/gpfs-nagios-info-sender.sh` is added to the cluster nodes via Quattor. This script sends the `mmlscluster`[7] command output directly to the Nagios server. In this way the Nagios side scripts can build configuration for all GPFS cluster, taking into consideration the quorum nodes and creating a cluster service for these particular machines.

To restore the server in case of disaster recovery comes into play Quattor. Quattor is able to reinstall the machine from scratch in about 30 minutes. Once the Nagios server is restored it can recreate his configuration: before the machines of the GPFS cluster must send their configuration with the script `/opt/gpfs-nagios-info-sender.sh` then by querying the Docet DB to know the hardware characteristics. Overall, the reconstruction of the configuration takes a few minutes. In Figure 1 we report the overview of the elements integration of the self-configuring control system.

4. The configuration system operation

Once the Nagios server has received the information of all the GPFS cluster, you can run the script that creates the configuration. There are two scripts that are executed in succession, the first is `/opt/nagios-gpfs/read_cluster_info.sh` and the second is `/opt/nagios-gpfs/create_all.sh`. The first reads the file sent by the GPFS cluster and creates an ordered list of servers per cluster, making sure that there are no machines belonging to more than one cluster, which is forbidden by GPFS. The second creates the configuration of Nagios itself, making sure that the servers are not in a blacklist, in which case they are ignored. It makes query to the database to check the hardware type of the individual server in order to add specific hardware controls (IPMI, RAID, POWER SUPPLY). Here are the steps that are followed:

(i) Control of information from the GPFS cluster and creation of a file containing the server.

(ii) Saving the current configuration in case of recovery.

(iii) Through requests to the Docet DB other types of machines (GridFTP, GPFS, etc.) are added, for these machines the “type” information is contained in docet.

(iv) It verifies that each server found in the preceding paragraphs is not present in the blacklist, in which case it is ignored.

(v) From the information gathered in the previous steps the script creates the configuration files of Nagios groups, hosts, commands and services.

(vi) A check is made of the correctness of the configuration with the command `nagios -v /etc/nagios/nagios.cfg`.

(vii) Nagios is ready to restart or reload the configuration.

For the farming department the steps are similar to the above with the exclusion of the first point since there is no GPFS cluster to control and that the machines which require information from Docet are Worker Node, User Interface and other server.

5. Some NagiosGraph test

In order to increase the number of information available from the Nagios server we have created some sensors to measure the parameters of the network and memory of the clients. These data are sent to the Nagios server which stores them in a Round Robin DB and displays them via the plugin NagiosGraph[8]. With a granularity of 5 minutes for each quantity kept for one year hundreds MB are filled. In Figure 2 an example of a NagiosGraph plot.

6. Conclusions

The automated system for Storage and Farming has been in production for over a year on two separate servers. During this period it has proven to be a reliable tool, which has significantly reduced the time spent managing Nagios and the possibility of manual errors. Possible developments are cross-checking of the two servers and in case of failure the automatic activation of a third Nagios that takes the place of the one server in case of hardrware failure.

References

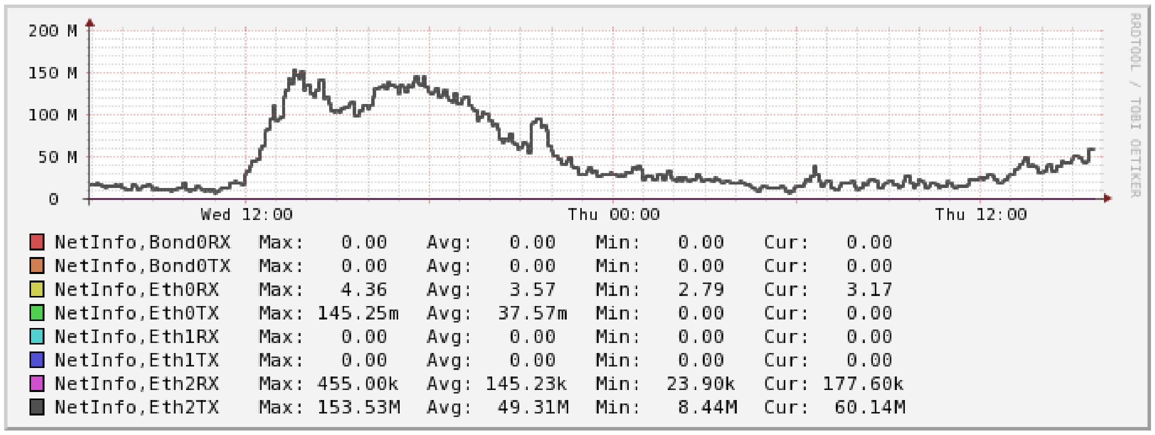
[1] D. Gregori et al, *INFN-CNAF Activity in the Tier-1 and Grid for LHC Experiments*, 2009, E-ISBN : 978-1-4244-3750-4

[2] `http://www.quattor.org/`

[3] `http://www.nagios.org/`

[4] S. Dal Pra, A. Crescente, *The data operation centre tool. Architecture and population strategies*, 2012, Journal of Physics: Conference Series 396(2012) 042014

[5] `http://exchange.nagios.org/directory/Addons/Monitoring-Agents/`
`/NRPE--2D-Nagios-Remote-Plugin-Executor/details`



```
🟥 NetInfo,Bond0RX   Max:   0.00    Avg:   0.00    Min:   0.00    Cur:   0.00
🟧 NetInfo,Bond0TX   Max:   0.00    Avg:   0.00    Min:   0.00    Cur:   0.00
🟨 NetInfo,Eth0RX    Max:   4.36    Avg:   3.57    Min:   2.79    Cur:   3.17
🟩 NetInfo,Eth0TX    Max: 145.25m   Avg:  37.57m   Min:   0.00    Cur:   0.00
🟦 NetInfo,Eth1RX    Max:   0.00    Avg:   0.00    Min:   0.00    Cur:   0.00
🟦 NetInfo,Eth1TX    Max:   0.00    Avg:   0.00    Min:   0.00    Cur:   0.00
🟪 NetInfo,Eth2RX    Max: 455.00k   Avg: 145.23k   Min:  23.90k   Cur: 177.60k
⬛ NetInfo,Eth2TX    Max: 153.53M   Avg:  49.31M   Min:   8.44M   Cur:  60.14M
```

Figure 2. Network NagiosGraph plot.

[6] http://exchange.nagios.org/directory/Addons/Passive-Checks/
/NSCA--2D-Nagios-Service-Check-Acceptor/details
[7] http://www-01.ibm.com/support/knowledgecenter/SSFKCN_3.5.0/
/com.ibm.cluster.gpfs.v3r5.gpfs100.doc/bl1adm_mmlscluster.htm
[8] http://nagiosgraph.sourceforge.net/