Available online at www.prace-ri.eu

Partnership for Advanced Computing in Europe

Design, Development and Improvement of Nagios System Monitoring for Large Clusters

Daniela Galetti[1], Federico Paladin[2]
SuperComputing Applications and Innovation Dept., CINECA, Bologna, Italy

Abstract

This document describes the work of design, development and improvement of the Nagios monitoring system done in Cineca and used for the Tier-1 systems participating in the PRACE projects. Starting from the issues arisen by the complexity of the HPC systems and the related monitoring activities, the targeted solutions and their implementation are explained. The most important aspects of the implementation and the specific issues related to HPC will be described with a specific attention to the exascale clusters.

1 d.galetti@cineca.it
2 f.paladin@cineca.it

Table of Contents

Introduction

Monitoring has always been an important part of the management system activities. With the increase of the number of server nodes of a computing cluster and the resulting raise of the probability of failures, monitoring is becoming even more important. The tools of monitor are very important to limit the impact of an infrastructure failure on the system productivity, especially in large scale production environments. They could let you spotting problems before they occur or at least reduce the intervention time and the time to solution. Between the different monitoring solutions, Nagios is one of the most open source solutions implemented in the IT field. After a technical analysis also Cineca choose Nagios as monitoring system for his services. After a specific project of development of custom scripts and specific configurations it became also an important part of the management of the High Performance Systems.

Nagios basic functionalities

The Nagios monitoring system3 provides a central view of the system status. Different dashboards provide at-a-glance access to monitoring information and views provide users with quick access to the information they find most useful. Alerts can be sent to management staff, business stakeholders, and end users via email or mobile text messages, providing them with details so they can start resolving process immediately. This information also could be used by organizations to plan for infrastructure upgrades before outdated systems.

A basic configuration of a monitoring system with Nagios is composed of a central server which collects the results of active and/or passive checks on different hosts and related services. Indeed for each host, you can define services that you want to monitor. The check of each couple (host, service) has a description and can assume 4 different values:

- OK (working status),
- WARN (warning status),
- CRIT (critical status),
- UNKN (unknown status).

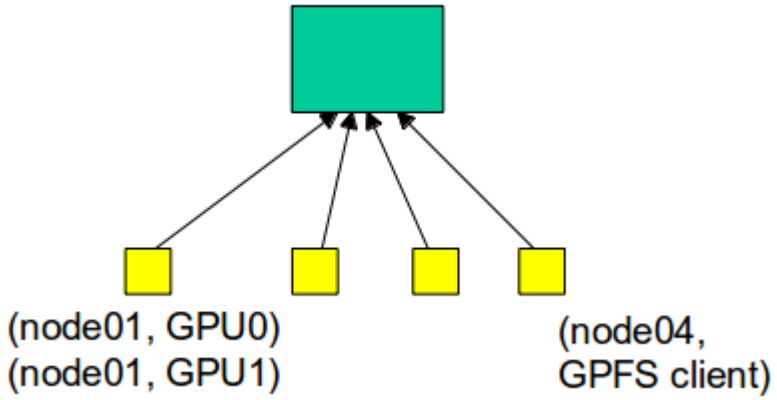

Figura 1: Nagios basic configuration

Nagios was designed to monitor in IT environments, with different hosts, connected with a site network, so the standard monitoring action are on a host/service base and organized in different views put together. This is why Nagios functionalities work well for a limited number of servers and/or a quite simple topology, but with HPC clusters that have many different parameters that have to be monitored on a large amount of nodes, it is better to take advantage of the flexibility of Nagios improving the monitoring tool with custom configuration and scripts. If you add to this that the amount of monitored elements are organized in a complex topology, for an efficient management of the entire system you will need more views, organized in different levels. That made necessary to move to a customized configuration and scripting of Nagios installation.

3 http://www.nagios.org/documentation

HNagios[4] (Nagios and HPC)

Because we need to monitor all the nodes of all our HPC clusters, with all their services, the first HPC issue that we faced was the number of the alarms to manage. To avoid Nagios server overload and improve its availability, the solution was: distribute monitoring in the time and in the space: executing different tests in different time frames and run checks on each monitored targets (called passive checks) was the first step of the customization. Also after this choices of implementation, the quantity of the signals sent to the server were still very big and it would be very difficult to spot critical situations and handle efficiently the different emergencies. So the way was specialize and organize views in different levels. We chosen to set up alarms in hierarchy and send to the Nagios server a critical summary with alarms and infos organized in a synthetic view that allows system administrators to distinguish between critical and non-critical events. In order to obtain this behaviour, we had to collect all the signals, organize and filter them before send them to the server. Thanks to a Nagios plugin developed by Mathias Kettner (MK Livestatus)[5] we could add a level of secondary Nagios servers (peripheral servers), each one running on the management node of each cluster, sending a structured summary to the central server.

In peripheral Nagios servers, we group nodes by their functionality in the cluster (management nodes, login nodes, visualization nodes, storage nodes, compute nodes, etc.) and define each group as host of the central Nagios server, so we can have only a view for each group of hosts. Regarding services, a service on a host of central Nagios instance is defined if and only if the same service is defined on at least one host of the referred group in the peripheral Nagios istance.

As the number of groups is much more little than the number of hosts, the information results more efficient.. Further, hosts on same group, having same primary functions in common, normally have same services associated to them. This improves the efficiency of the views. If in peripheral Nagios servers there is a service with different status on nodes, the worst service status is sent to the central server, with a count of the nodes where the service is in that status.

To avoid a lost of information from the peripheral Nagios, we found useful to add the status "-" (minus) to the 4 standard Nagios status of a host/service. This new status describes the situation of a peripheral host with a failure, that is already managed and the alarm was acknowledged.

- OK (working status),
- WARN (warning status),
- CRIT (critical status),
- UNKN (unknown status),
- - (already managed critical status)

This customized status is based on the "*scheduled downtime*" Nagios concept and could be managed from command line.

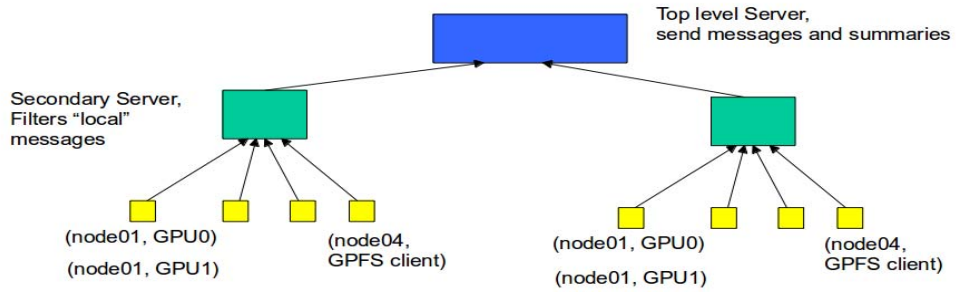

Figura 2: HNagios hierarchical configuration

Finally, the central Nagios reports, in the description of the service, hostnames and counts of the peripheral nodes with the service in the same state, ordered by decreasing impact (first critical, then warning, last already managed). Further, in case a status counts a single node, the alarm contains also a brief description of the problem (from the peripheral Nagios).

[4] https://hpc-forge.cineca.it/files/hpc-sysmgmt/public/hnagios/hnagios.tar.gz
[5] http://mathias-kettner.de/checkmk_livestatus.html

For example, if a user job exhausted the memory of 200 compute nodes...and there are also a warning on one node and 3 nodes on maintenance:

- Standard Nagios sends 100 emails
- HNagios sends only one email with a line like this:

```
Mail Subject: PLX-compute(R-memory::swap::free) ==> CRITICAL
Mail Body:
           C(100):node[054,131-136,143-154,133-208,231-235]
           W(1): node[0002]{KO: 20032654 < 40% (wrt 50331636 KB)}
           -(3): node[014-016]
```

Conclusions

All this customized changes combined together, allows us to distinguish in a glimpse between more and less critical status without loss of information. Indeed if we need more details about an alarm sent by the central Nagios server, for example for debugging purpose, we can find it in the peripheral Nagios server. The consequence is an easier management of the priorities and organization of the repair service in a way suitable for the whole production of the computing system, modulating the time reaction in each event. We started to think about this problem when we moved to hundreds of nodes to thousands of managed node, but we completed the new monitoring infrastructure when there were already tens of thousands of nodes in the machine rooms, and it is still working well. We think that this configuration with this changes is very scalable and it would need at least some more little changes for different topologies or node hierarchies, but it will very usable also with exascale dimension.

Acknowledgements

This work was financially supported by the PRACE project, funded in part by the EUs 7th Framework Programme (FP7/2007-2013) under grant agreement no. RI-211528 and FP7-261557.5