Monitoring a HPC Cluster with Nagios

Piero Calucci
Scuola Internazionale Superiore di Studi Avanzati
Trieste

2009-04-03

1 Try again… Fail better.

Outline

1 Nagios Concept
2 Nagios Web Interface
3 Nagios Installation for HPC Monitoring @SISSA

What is Nagios?

«NAGIOS® is a system and network monitoring application. It watches hosts and services that you specify, alerting you when things go bad and when they get better». — Nagios documentation

Architecture

All the hard work is done by plugins, the nagios daemon «only» schedules them to be executed at the right time with the right parameters and collect results.
The cgi interface is entirely optional, but highly useful.
The **nagios daemon**
- schedules and executes **active** host and service checks
- accepts asynchronous **passive** checks
- sends out **notifications** on host or service state change
- executes **event handlers** on host or service state change
- writes and rotates **log** and **state** files

**Local Plugin Execution**

All **active checks** involve the local execution of some plugin.

Locally executed plugins can just check for some local service...
NRPE allows execution of plugins on remote hosts. Remote plugin results are reported to Nagios by the locally executed check_nrpe plugin.

The NSCA daemon relays to Nagios asynchronous notifications sent by send_nsca (this is how passive checks work).

**Host Checks**
- a **host** is basically anything that can be given a name and an address
- hosts can be **UP**, **DOWN** or **UNREACHABLE** (the host may well be up and running, but something in the network in between is broken)
- host checks are executed
  - at regular intervals
  - on-demand when a service on the host changes state
  - on-demand when required by reachability or dependency logic

**Service Checks**
- a **service** is any monitored «thing» associated with a host
- service state can be **OK**, **WARNING**, **CRITICAL**, **UNKNOWN**
- service checks are executed
  - at regular intervals
  - on-demand when required by dependency logic
State Types

Each host or service state can be of **SOFT** or **HARD** type. SOFT type states are considered «uncertain» or «transitioning» and are checked with a different (usually higher) frequency until a specified maximum retry count is reached – they then become HARD states.

Flap Detection

- When a host or service changes its state «too frequently» it is detected as being **flapping**
- Flapping hosts do not trigger notifications in order to avoid filling up mailboxes
- Flap detection threshold is configurable, and flap detection can be disabled entirely. However, the defaults are good enough

Notifications

- notifications can be sent out whenever a HARD state transition occurs and when a host or service remains in a non-OK hard state for a specified time
- notification can enabled or disabled for each host or service
- notification times and contact groups can be set up so that only the right person is contacted, only when he is on duty

Event Handlers

- event handlers are executed (if defined) when a host or service changes state and for each retry in SOFT states
- they are given all state information: state, type, retry count
- they can do basically anything, as long as they are given sufficient permissions, including
  - restarting a failed service or host
  - changing nagios configuration by writing to the command pipe (adaptive monitoring)
Monitoring a HPC Cluster with Nagios

Piero Calucci

Nagios Concept
Web Interface
HPC

Tactical Overview

- Hosts
  - 4 Down
  - 0 Unreachable
  - 164 Up
  - 0 Pending

- Services
  - 27 Critical
  - 3 Warning
  - 0 Unknown
  - 1392 Ok
  - 91 Pending

- Monitoring Features
  - All Services Enabled
  - All Services Disabled
  - All Hosts Enabled
  - All Hosts Disabled

Service Status Detail

- Current Status:
  - Host State: Down
  - Status Information: No status available
  - Performance Data: No performance data available
  - Current State: Down
  - Next Scheduled Action: None

Service Problems

- Service Status Details for Hosts
  - Host Status: Down
  - Last Update: 03-26-2019 12:51:46

Host State

- Host Information:
  - Last Updated: 03-29-2019 12:51:24
  - Check Type: ACTIVE
  - Check Duration: 0.590 / 4.276 seconds
  - Last State Change: 03-26-2019 12:54:44

- Host State Information:
  - Host Status: DOWN
  - Status Information: No status available
  - Performance Data: No performance data available
  - Current State: Down
  - Next Scheduled Action: None
  - Last Update: 03-26-2019 12:51:46

Member of: nc-nodes

10.2.10.7

Service Status Details for All Hosts

- Host Status:
  - Host Name: node1
  - Status: Down
  - Last Updated: 03-26-2019 12:51:46

- Service Status Details for Hosts:
  - Host Status: Down
  - Last Update: 03-26-2019 12:51:46
What to Monitor

Hosts

- a masternode
- 160+ computing nodes
- several NFS servers
  - including a HA NFS cluster
- lustre servers

What to Monitor

Services

- «generic» services (SSH, NTP, . . .)
- HPC-specific services (maui, pbs_server, pbs_mom)
- computing node health (load average, hardware errors, . . .)

What to Do

- send notifications when things go unrecoverably bad
- avoid sending out notifications every hour (Nagios default) – how often is too often?
- restart services and hosts when possible
  - how much does it take to declare a service crashed or a computing node dead?
  - do we trust Nagios to detect correctly?
  - do we accept the risk of rebooting a node that was just responding late?

Open Issues

- «undead» nodes
  - what are we going to do with nodes that reply to ping but nothing else?
    - (they are UP from Nagios PoV)
  - how are we going to reliably detect the undead state?
- host reachability
  - cluster network topology is too simple to make Nagios reachability logic useful (no pingable gateways)
  - we have no way to check single switch port (do we?)
- multihomed nodes
  - Nagios has minimal support for multihomed hosts (or maybe I didn’t understand it . . .)
  - we have no clear way to know all addresses associated with a node FQDN
Open Issues
High System CPU Time

After 3 weeks we are over 30% CPU usage on a 2x dual core opteron...

Example
Check pbs_mom /1

```plaintext
define host {
    use       linux-server
    host_name p001
    address   10.2.13.1
}
```

```plaintext
define hostgroup {
    hostgroup_name p-nodes
    alias         planck nodes
    members       p001,p002,p003,...
}
```

Open Issues
File Access Permissions Mess

On monitoring server:
- nagios needs to read its own configuration files and write log and status files
- web server needs to read some nagios config files and write to nagios command pipe
- nagios event handlers need to write to nagios command pipe
- nsca needs to read its own configuration and write to nagios command pipe

On monitored hosts:
- nrpe needs to read its own configuration and execute nagios plugins
- send_nsca needs to read its own configuration

Example
Check pbs_mom /2

```plaintext
define service {
    use       generic-service
    hostgroup_name m-nodes,c-nodes,p-nodes,...
    service_description pbs_mom
    check_command check_pbs_mom
    max_check_attempts 4
    notifications_enabled 1
    event_handler restart_pbs_mom
    servicegroups sg_node_batch
}
```
Example
Check pbs_mom /3

```python
define command {
    command_name check_pbs_mom
    command_line $USER1$/check_tcp -H $HOSTADDRESS$ -p 15002
}
```

Example
Check pbs_mom /4

```python
restart_pbs_mom.sh is:
#!/bin/sh

SSH="ssh -i /var/lib/nagios/.ssh/id_dsa -t -t"
RESTART="sudo /sbin/service pbs_mom restart"

case $2 in
    CRITICAL)
        case $3 in
            SOFT)
                case $4 in
                    3)
                        $SSH $1 $RESTART
                    ;;
                esac
            HARD)
                $SSH $1 $RESTART
            ;;
            esac
        ;;
        esac
    ;;
    esac
```

That's it

This slide intentionally left blank