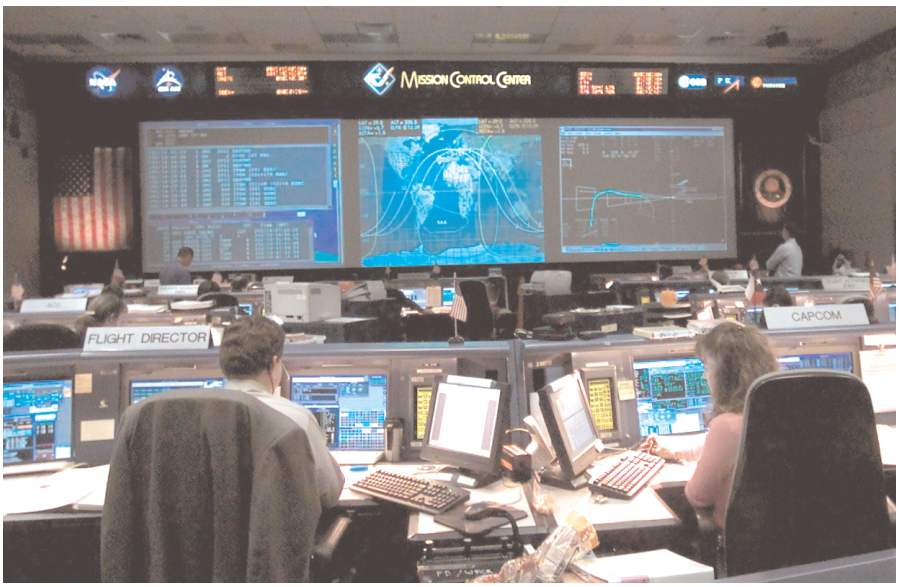
Network Management with Nagios, Netsaint's Successor

What's Going On?

Nagios is network monitoring software that will help admins keep an eye on their networks. It is the successor to Netsaint's and is now easier to configure, adds new features and an improved architecture. BY DIETMAR RUZICKA



Knowing what is going on in a network is a most critical task for most systems administrators. Has a computer just failed, is a filesystem full, has a service just crashed? The admin should be the first to know, and quick to remedy the situation. Network monitoring programs like Nagios [1] can help. Ethan Galstad released Version 1.0 late in 2002 – and just like its immediate predecessor, Netsaint, Nagios was released under GPL. The new name is based on the Greek word, hagios (saint).

Nagios inherits the features and open architecture of its predecessor, adds some functionality and a more simple, template based configuration. In the short time since the programme was released, Nagios has proved its value in at least one large installation (see the "Nagios in the Austrian Salt-Mines" box).

Nagios comprises several components (see Figure 1). The core component, referred to by the documentation as the Nagios Process or Core Logic, is the central process. It uses plug-ins [4] to collect information, parses the data, and writes to log files. When the central process recognizes a problem, it sends a message to the administrators.

The Web interface and its CGI scripts read the information collated in files and render it in a browser. You can use a named pipe, referred to as an external command file, to pass additional commands to the Nagios process. This interface, which is also Web based, writing commands to the pipe file which are read and executed by Nagios.

Plug-in Architecture

The Nagios processes call plug-ins to collect information on the status of hosts and services. These plug-ins can be arbitrary scripts or compiled programs, provided they implement the Nagios interfaces. The status of the query is itself contained in the return code of the plug-in; see Table 1. Additionally, Nagios reads the first line of the plug-in output ("stdout"), stores the information it has read in the log files and, if required, notifies the admin.

Quicker Plug-ins thanks to Embedded Perl

Nagios can optionally be compiled with the EPN interpreter; the complete system is then referred to as Embedded Perl Nagios. The EPN interpreter calls Perl plug-ins very efficiently: it uses a library call instead of the fork exec process normally required, and thus does not need to load the Perl interpreter every time.

Unfortunately, some Perl plug-ins will not work with EPN, and debugging facilities are restricted. The "contrib" directory provides the "mini_epn" program, which is a lot handier for testing plug-ins than a full-blown Nagios system.

The Nagios process is configured using text files, the central files being "nagios.cfg", "resource.cfg", and "cgi.cfg". "nagios.cfg" binds additional files that contain object configurations for hosts, services, and contacts (see Table 2). A new feature allows the admin to bind complete directories, including the files they contain, to Nagios:

```
cfg_file=/etc/nagios/hosts.cfg
cfg_dir=/etc/nagios/hosts
```

Formerly (that is, with Netsaint), extended information on hosts and services had to be entered directly in the "cgi.cfg" file. Nagios allows you to swap the "hostextinfo" and "serviceextinfo" configuration out to individual files, which are then bound by "cgi.cfg". The syntax of these files is a lot clearer than the old method.

The administrator defines objects to represent a network with its hosts and services in Nagios, specifying the host for each service (for example an NFS export or mail server). Individual hosts can be combined to form a "hostgroup"; a host can be a member of several groups. This makes it easy to assign a service to our group of hosts.

Structured Configuration

The "parents" directive allows the admin to define the structure of the network when defining a host. This is very useful because, if a router that connects the Nagios server to another subnet goes down, Nagios simply reports a router failure. The system then marks any hosts that have this router defined as a parent as unreachable.

The administrators of monitored systems are also grouped as the "contact-group" in the Nagios configuration. An admin can belong to several groups. Nagios sends system status messages to the groups. If the Web interface requires authentication, contacts are also used as user names. The Web front-end automatically hides any hosts from which the authenticated contact should not receive messages. Authentication is also necessary if you want to use the Web interface to influence the Nagios process, such as preventing it from checking a service for a short time.

Nagios checks hosts and services periodically. The "timeperiod" specifies the times to perform tests or send messages to contacts. This feature is extremely useful in real-life situations, as an admin will often not care whether a service is reachable or not outside office hours. Having said that, there are some services (such as the Web service) that the admin needs to look into right away.

To allow Nagios to run external commands, the admin needs to define objects for these commands in the configuration file, where "command" describes the command line including any options and parameters. Commands are mainly used for testing hosts and services, but they may also handle events, or even transmit mail or short messages to the admin.

Nagios has two types of component states soft states and hard states, which can be defined as up, down or warning. This allows the program to differentiate between real problems and temporary malfunctions. A change in status can initiate various actions, depending on whether it occurs in a soft or hard state. If the plug-in notices a host service failure, it will first change to the appropriate soft state. If, after performing additional tests, Nagios ascertains the status of "OK" before the "max_check_attempts" threshold has been reached, Nagios terminate the soft state without generating a message. A real problem has not occurred in this case.

If too many tests fail, the status type changes to hard state, and this prompts the Nagios process to initiate messaging routines (who gets what message?) and run the registered event handler.

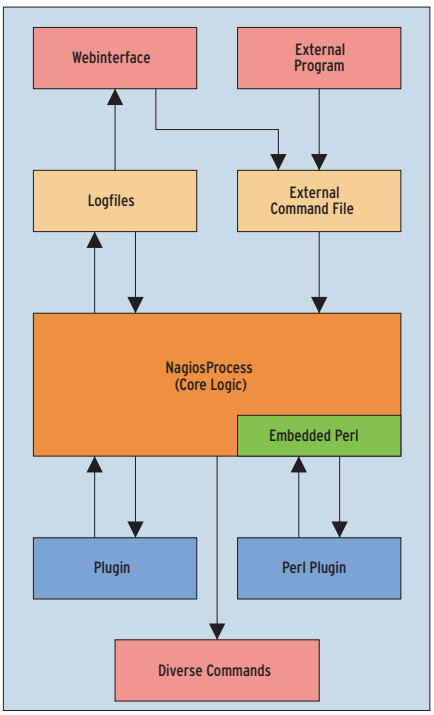


Figure 1: The Web interface and other external programs use External Command Files to control the Nagios process. Nagios itself uses plug-ins to monitor hosts, for example

Nagios at the Austria Salt-Mines

The Austria Salt-Mines [2] are one of the oldest industrial companies in Europe; salt has been mined in the Austrian Salzkammergut region since pre-historic times. Today modern 24x7 technologies replace traditional skills.

The mining equipment is controlled by a high availability IT network with 750 nodes, comprising a total of 100 servers that offer almost 1000 services. The Austria Salt-Mines gave Vienna based Cubit IT the job of installing Nagios in this environment to provide non-stop network monitoring.

New Plug-ins

The task of detailed monitoring of Oracle instances, run by the "check_oracle" plug-in, has clearly exceeded the capabilities its original job description. Three new plug-ins have been developed to resolve the issue: "check_oracle_sql", "check_oracle_tablespace", and "check_oracle_tablespace_status". They require a database account to read meta-data. The plug-ins are also capable of remotely checking Oracle.

The three Perl scripts connect using the Perl DBI (Database Interface) to contact the database and run SQL statements. In the case of "check_oracle_sql", the "SELECT user FROM dual" statement checks whether the instance is running correctly. "check_oracle_tablespace" checks the space usage in the table, using system view to do so. "check_oracle_tablespace_status" returns the tablespace status.

A Linux Heartbeat cluster [9] with a DRBD (Distributed Replicated Block Device, [10]) is used for final production control (salt packaging) and as a database server at the salt-mines. The new "check_drbd" plug-in checks the status of the DRBD by reading the /proc filesystem entries.

New plug-ins also monitor the enterprise wide Novell Groupwise mail system. Some status data can be retrieved via SNMP (Simple Network Management Protocol), although the "check_aktiv_email" plug-in performs more exhaustive tests. This sends a mail message to an external host, which autoreplies. The plug-in then uses POP3 to accept the mail back into the Groupwise system, thus ensuring that email exchanges with the rest of the world still work.

Checking network connectivity was a real challenge. The salt-mines use a Novell Border Manager as a proxy, and the system requires regular user authentication. Lynx provided a solution for this challenge: the "check_bordermanager" plug-in now attempts to load a few national and international pages.

Nagios monitors the whole salt-mine network from this room

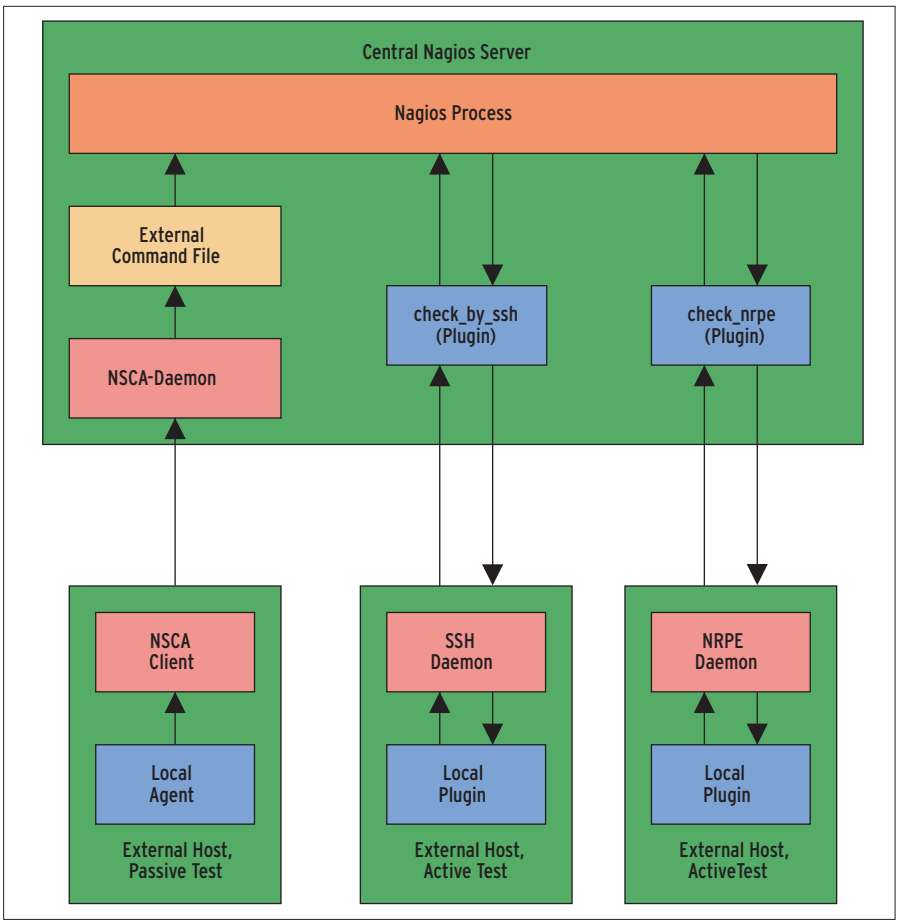


Figure 2: Nagios has three ways of running tests on external systems: actively using SSH or NRPE (Nagios Remote Plugin Executor), or passively using NSCA (Nagios Service Check Acceptor). The passive variant involves the host reporting events

Testing Hosts and Services

Nagios distinguishes two types of service test: active and passive. The Nagios process launches plug-ins for active tests at regular intervals, as defined in "normal _check_interval". The plug-in polls the status of the service or host.

A plug-in running locally on the Nagios server can only check the external behavior of a host. Ethan Galstad has programmed an additional module, called the NRPE (Nagios Remote Plugin Executor), to run plug-ins directly on external hosts. The Nagios server uses the "check_nrpe" plug-in to communicate with the NRPE daemon on the monitored host. Alternatively, Nagios can call the "check_by_ssh" plug-in to use SSH to launch a program on the host.

In the case of passive tests, Nagios expects an agent to report its findings using an "External Command File". Passive tests are particularly useful for asynchronous events, such as SNMP traps. Agents can transfer data generated by events of this kind to the Nagios server using the convenience of the NSCA (Nagios Service Check Acceptor). The client-side "send_nsca" then sends its results to the NSCA daemon which is running the passive command of the Nagios process.

The freshness check is another of Nagios' new features compared to Netsaint. In the case of passive tests, Netsaint cannot determine whether an agent has simply stopped working or if a problem simply has not occurred for awhile.

This issue is resolved by the "freshness_check": if the Nagios process has not had a message from an agent within the defined period, it will initiate an active test.

Scheduled Downtime

High availability networks need to plan downtime, and Nagios needs to know when this is, to avoid unnecessary alerts. Sooner or later the system will need some maintenance or an upgrade. Systems that do not need to be available 24x7 can be shut down outside of normal working hours, with high availability systems having to wait for a scheduled downtime window.

Another new feature in comparison to Netsaint is the fact that Nagios can choose between flexible and fixed downtime. Whereas the latter defines a fixed starting and endpoint, a flexible downtime simple defines a limited period during which the host (or service) does not have to be available. If the host fails during a flexible downtime period, Nagios will wait for a pre-defined period before re-checking the host.

Table 1: Plug-In Return Codes

Return Code	Status	Meaning
0	OK	Test completed successfully, the service works.
1	Warning	Test completed successfully, but the result was outside tolerance.
2	Critical	Test did not complete successfully, or the result was critical.
3	Unknown	The plug-in was unable to perform the service test; the result is ambiguous.

Table 2: Configuring Nagios Objects

Object	Description
host	Defines a server, a workstation, etc.
service	Describes a service (HTTP, NFS etc.) provided by the server.
contact	Nagios informs this person in case of emergency.
hostgroup	Collates multiple "host" entries with the same characteristics in a group. Each "host" must belong to at least one host group.
contactgroup	Groups people to allow sending messages to them simultaneously.
timeperiod	Defines the times when a "host" or "service" can be tested or a "contactgroup" should be informed.
COMMAND	Describes how to launch plug-ins and other external tools.
host/service dependency	Defines the dependencies for "host" and "service"
host/service/hostgroup escalation	Defines the notification escalation procedure.

Web Interface

Nagios provides a comprehensive Web interface offering an overview of the current network status. In addition to the Tactical Overview (Figure 3) which provides information on the Nagios process, by showing the service and host tests planned next, with any comments entered by admins and the next scheduled downtimes. The comprehensive reporting facilities play an important role, and you can also view transmitted messages, the event log, and the configuration.

The Status Summary displays the status of the complete system in a neat table. The Status Map (see Figure 4) provides various views of the network structure and shows the host dependencies. This view only makes sense for smaller networks, however, because it soon becomes confusing.

The Alert Summary is new, and helps the admin quickly discover vulnerable servers. The Alert Histogram is another new feature that shows accumulations of problems. Admins can also manipulate the Nagios process via the Web interface, using it to enable or disable individual tests, add scheduled downtime or comments, or re-start processes.

Alerts on the Network

In cases of alerts, administrators normally prefer to be notified rather than open the Web interface to discover what has happened. Nagios reflects this by generating a notification whenever a hard state changes. Special filters prevent the admin from being bombarded with thousands of messages.

The first filter level allows you to enable or disable notification globally. The second stage provides host and service filters divided into four different levels. Additionally, Nagios will not notify the admin when:

- downtime is scheduled for the host or service,
- a component keeps oscillating between two states (flapping),
- notification has been disabled globally for a component, or
- the problem occurs outside of the notification period.

The last filter level comprises contact filters. The admin can define the status that Nagios will have to reach before notify each user of, say warning or critical. These definitions can be based on the hosts and/or services. This level also allows the admin to define notification times for the user.

The Right Contacts

Nagios uses contact groups to find out what admin it should notify for a specific service or host group. Nagios prevents an admin receiving identical messages more than once. There are no restrictions to the channels used for alert messages. Email, SMS or instant message services, such as ICQ, are typical candidates. In the case of the salt-mines, both email and Nokia GSM 20 [5] in combination with gsmlib [6] have proved to be most valuable: Nagios uses these resources to send a message, by SMS or phone, to the admin.

Nagios provides multiple alert levels (notification escalation), allowing for unrestricted definition of notification

Templates and Inheritance

The most important new features in Nagios compared with Netsaint are the program's object oriented configuration methods – templates, which means admins only having to type repeated parameters just once. Templates are available for every service and host specific setting. They use three new variables:

```
define objecttype{
   name         Template name
   use          Name of parent
template
   register     [0/1]
   # object specific definitions
}
```

The "name" variable defines a unique identifier for an object. Other objects can call the "use" directive and refer to this object by name, to inherit the settings defined in the template. "register" specifies whether this object is real, and will thus be seen by the Nagios process ("1"), or an abstract object ("0") that is used only as a parent for other objects.

As is typically the case in inheritance, local definitions will override inherited characteristics:

```
define host{
   host_name            host1
   check_command        check-host
   notification_options d,r
   max_check_attempts   5
   name                 template1
}
define host{
   host_name            host2
   max_check_attempts   3
   use                  template1
}
```

"host2" calls "use template1" to inherit the characteristics of "host1". It overwrites the value for "max_check_attempts" with a value of "3", instead of "5" for "host1". A well defined template structure greatly simplifies the configuration, especially if it affects several similar hosts and services. Thus a new host can be monitored by adding a few lines to the configuration.

Listing 1: Example of a contact definition

```
01 # generic contact definition
02 define contact{
03    name                            generic-contact
04    service_notification_period     24x7
05    host_notification_period        24x7
06    service_notification_options    w,c,r #warning,
critical, recovery
07    host_notification_options       d,r #down, recovery
08    service_notification_commands   notify-by-email
09    host_notification_commands      host-notify-by-email
10    register                        0
11 }
12
13 # 'nagios' contact definition
14 define contact{
15    use                             generic-contact
16    contact_name                    nagios
17    alias                           Nagios Admin
18    email                           nagios
19 }
```

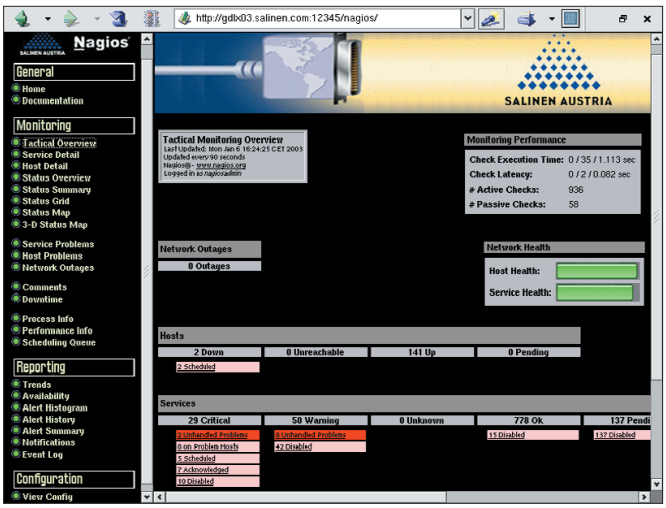


Figure 3: Nagios uses the Tactical Overview to show an overview of the system status. The area in the red frame shows that two hosts are down. In this case the downtime was scheduled

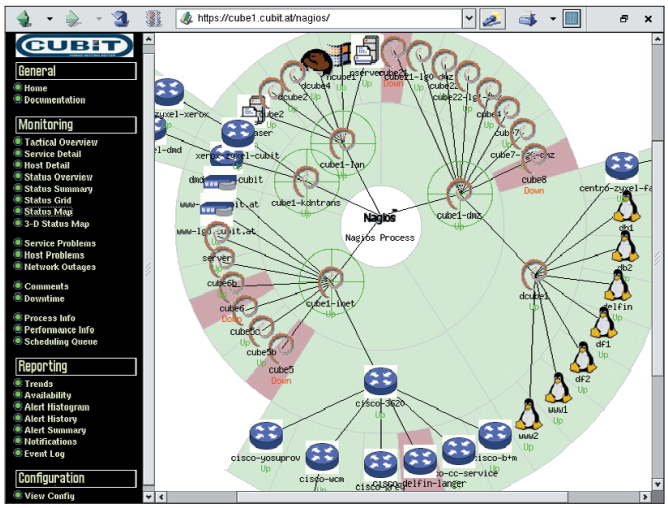


Figure 4: The Status Map uses the Circular View to provide an overview of the network structure. The parent relationships of the various hosts are shown in a tree structure

processes. Normally, that is without escalation, Nagios sends a single alert in case of a problem and repeats the message after a defined interval. The system administrator can define the messages, and intervals for more urgent cases.

Notification Escalation

The salt-mines use notification escalation for a few critical systems (see Listing 2). In case of failure in the final production plant, Nagios not only transmits a standard alert by email, but sends a short message for the first three alerts to the system administrator in charge. If the administrator fails to react within this period, the system calls the admin's cellphone every 15 minutes. This takes care of situations where short messages get lost or the on-duty admin does not notice that a message has arrived.

Notification escalation can also be used for event handling. At first glance this may seem a slightly roundabout way of doing things, but it does provide flexibility. Nagios, in this mode, can first attempt several automatic responses to an issue, before alerting an admin at home. A normal event handler only has one shot, in contrast.

Clever Use of Escalation

The idea is to create a pseudo-contact. A "contact" definition includes both "service_notification_commands" and "host_notification_commands". Instead of dispatching an e-mail or short message, the admin can define additional actions, such as rebooting the Web server. This method is admittedly complex, as the admin needs to define both the command, and additional objects, such as the "contact_group", a "contact" and a "serviceescalation".

If you do not need escalation levels, you might prefer to stick to simple events. You can use events as triggers that refer to "command" definitions to launch arbitrary programs. Nagios calls the event handler when a service or host fails, when a host enters a soft fail state, changes from a soft to a hard state, or reverts to a normal state.

In large networks it is often necessary to distribute network monitoring to several servers at several sites, but at the same time collate this data in one central

Listing 2: Escalation Notification Levels

```
01 # AS/400 everything
02 define serviceescalation{
03     host_name               susen2
04     service_description     *
05     first_notification      1
06     last_notification       3
07     contact_groups          admins, notify-by-sms
08     notification_interval   5
09 }
10 define serviceescalation{
11     host_name               susen2
12     service_description     *
13     first_notification      4
14     last_notification       0
15     contact_groups          admins, notify-by-call
16     notification_interval   15
17 }
```

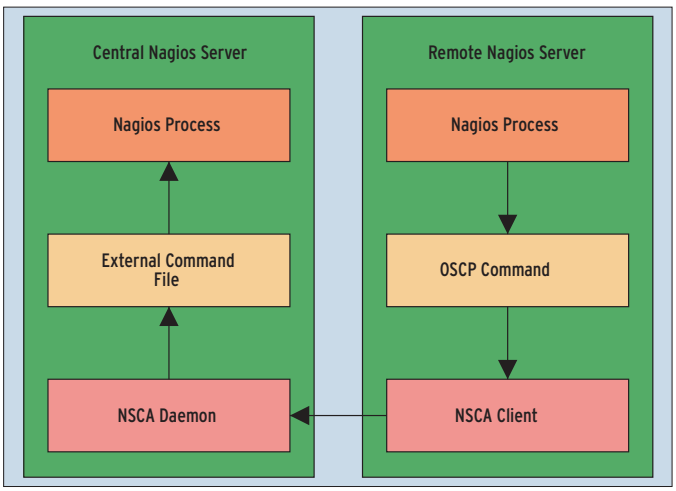


Figure 5: Distributed monitoring requires servers to report status information to the central monitoring server via the OSCP command (Obsessive Compulsive Service Processor) and NSCA (Nagios Service Check Acceptor)

repository. Nagios provides suitable facilities. The central server has the same features and components as a single server, whereas the distributed servers are restricted to the Nagios process and its plug-ins.

Distributed Monitoring

The distributed servers and the central server use the NSCA to communicate. The services running on the distributed servers are defined as passive tests on the central server. Hosts cannot be tested in this way, even though the central server still needs to perform an active test.

The "obsess_over_services" directive is responsible for passing information from distributed servers to the central repository. If this facility is active, Nagios runs the Obsessive Compulsive Service Processor Command for each service test, following the event handler and notification. The command can be defined to pass information to the NSCA client, which uses the NSCA daemon to pass the information to the Nagios process.

In typical heterogeneous environments you tend to encounter problems that might be difficult to solve without resorting to a few tricks. Nagios' open structure is a big help in this case.

One of the stipulations set by the salt-mines for Nagios was that the network monitoring system should include the final production and the automatic forwarding systems. Although these systems use a Linux cluster as an application and database server, the clients themselves run on Windows, and there are even some Netware servers in there.

All of these systems require different agents to interact with Nagios. The most critical agents perform log checking and query applications to discover if they are still alive.

Log Checks on Multiple Platforms

The log check uses a cronjob to check the log files at regular intervals, testing for regular expressions and reporting the status to Nagios. The activity check tests whether a monitored application regularly modifies files, and raises an alerts if not. Both programs are implemented in Perl; the interpreter is available for all three platforms.

The problem was getting the log check the send its results to the Nagios process in the form of a passive test. This is trivial on Linux; NSCA provides everything you need. Windows needs to take a little detour: the agent uses OpenSSH (in the Cygwin version) to launch the "send_nsca" command remotely on the Nagios server. OpenSSH also provides for active management of the Windows hosts: if required, Nagios can log on to the Cygwin SSH server and reboot a computer, for example.

The range of possibilities was far more restricted on Netware. Here, the local agents uses e-mail to send its results to the central server, which in turn, evaluates the message and then uses an external command file to pass them on to Nagios.

To be Announced

A few useful features are in the pipeline for the next major release of Nagios, some of them have already been implemented. One of the functions that has already been set in place is the performance passive hosts testing, which greatly simplifies distributed monitoring.

If you have to revise your Nagios configuration regularly, you will appreciate the fact that the Web interface only works as expected if your configuration is consistent. Version 2 will use a Cached Object Definition File to prevent problems if configuration files are modified after launching Nagios. The use of regular expressions should help save the admin a lot of typing in future.

Nagios 2.0 will be dropping quite a few older features. For example, it will no longer be possible to enter our checks definitions and advanced host information directly in the "nagios.cfg" file, although this method was available both in Netsaint and Nagios 1.0. A more flexible template based solution will then permanently replace the older variant.

Version 3 plans to exchange CGIs - programmed in C PHP Web interface. Status files, comments scheduled downtime data are currently stored in three log files, although Ethan Galstad with currently considering placing them in an XML formatted file. ■

Integrating Other Operating Systems

In most large networks, admins will be expected to take care of one or two Windows servers, Netware servers, an AS/400 or other operating systems. An operating system specific client is required to allow Nagios to monitor these systems. In the case of Windows this is the NS Client (Netsaint Windows Client) by Yves Rubin [7]. It allows you to monitor the CPU load, memory and hard disk usage, the status of various services and processes, and many other things. The NS client must be installed on the Windows host; the information it produces being read by the "check_nt" plug-in.

James Drews' [8] MRTG extension provides an extremely useful Netware client. The Netware extension allows you to monitor the CPU load, volume usage, the amended thread count, various caches and buffers, the status of the DS database, login status (enabled or disabled) and many other things. To do so, you simply load the "MRTGEXT.NLM" module on the Netware host, and allow Nagios to collect the data using "nwstat.pl" or the "check_nwstat" plug-in.

INFO

[1] Nagios: *http://www.nagios.org*

[2] Austria Salt-Mines: *http://www.salinen.com*

[3] Cubit IT: *http://www.cubit.at*

[4] Nagios Plug-ins: *http://nagiosplug.sourceforge.net/*

[5] Nokia Products for Business: *http://www.nokia.com/nokia/0,5184,2970,00.html*

[6] Gsmlib: *http://www.pxh.de/fs/gsmlib/*

[7] NS Client: *http://nsclient.ready2run.nl/*

[8] MRTGEXT.NLM: *http://www.engr.wisc.edu/~drews/mrtg/*

[9] Linux-HA, Heartbeat: *http://www.linux-ha.org/*

[10] DRDB: *http://www.complang.tuwien.ac.at/reisner/drbd/*

THE AUTHOR

Dietmar Ruzicka is a computer science student at the Technical University in Vienna, Austria. He is responsible for Vienna based Open Source service provider, Cubit IT's, Nagios project.