Assessing Groundwork Monitoring Server Performance

Contents

Introduction

This document is intended to help you assess your GroundWork Monitoring Server's performance. As the server's workload increases towards its upper range, certain functions will begin to manifest problematic behavior.

Typical First-Noticed Symptoms

1. One typical symptom of a over-loaded server is gaps in RRD Performance Graphs. As check latencies increase, data points may start falling outside the allowed tolerances and be dropped from the graphs.
2. Changes made in the Monarch 'Configuration' Tab take noticeably longer to propagate from a Monarch 'commit' to the GroundWork Status Viewer pages.

Nagios Engine Program-Wide Performance Information (Check Latency)

The first place to look for indications that your system is over-loaded is the Nagios Performance CGI. From the GroundWork Portal, click on 'Monitoring Server' option. This will bring you to a Nagios display of the Monitoring Server and its services. In the secondary menu, you then pick 'Performance Information'.

The most likely metric to show you any performance load on the server is the Active Service Check Latency metrics (Min,Max,Avg) in the top right table shown. These metrics are in seconds. 180 seconds is three minutes, which isn't bad. But 1800 seconds is thirty minutes, which *is* bad.

Check Latency is the time difference between when a service check was scheduled to run and when it actually ran.

Nagiosstats

Nagiosstats is a command-line version of the CGI shown above. It is simpler and has less dependencies. It also carries slightly different information, such as 'Total Services' and 'Total Hosts'

```
[root@lunias ~]# $GW_HOME/nagios/bin/nagiostats
Nagios Stats 2.5
Copyright (c) 2003-2005 Ethan Galstad (www.nagios.org)
Last Modified: 07-13-2006
License: GPL

CURRENT STATUS DATA
----------------------------------------------------
Status File:                          /usr/local/groundwork/nagios/var/status.log
Status File Age:                      0d 0h 0m 5s
Status File Version:                  2.5

Program Running Time:                 3d 3h 20m 51s

Total Services:                       1417
Services Checked:                     1289
Services Scheduled:                   1417
Active Service Checks:                1417
Passive Service Checks:               0
Total Service State Change:           0.000 / 80.660 / 0.214 %
Active Service Latency:               213286.359 / 215290.335 / 214727.280 %
Active Service Execution Time:        0.000 / 52.552 / 10.106 sec
Active Service State Change:          0.000 / 80.660 / 0.214 %
Active Services Last 1/5/15/60 min:   0 / 0 / 0 / 0
Passive Service State Change:         0.000 / 0.000 / 0.000 %
Passive Services Last 1/5/15/60 min:  0 / 0 / 0 / 0
Services Ok/Warn/Unk/Crit:            187 / 23 / 668 / 539
Services Flapping:                    0
Services In Downtime:                 0

Total Hosts:                          260
Hosts Checked:                        260
Hosts Scheduled:                      0
Active Host Checks:                   260
Passive Host Checks:                  0
Total Host State Change:              0.000 / 8.160 / 0.031 %
Active Host Latency:                  0.000 / 0.000 / 0.000 %
Active Host Execution Time:           0.029 / 17.250 / 9.365 sec
Active Host State Change:             0.000 / 8.160 / 0.031 %
Active Hosts Last 1/5/15/60 min:      0 / 0 / 0 / 0
Passive Host State Change:            0.000 / 0.000 / 0.000 %
Passive Hosts Last 1/5/15/60 min:     0 / 0 / 0 / 0
Hosts Up/Down/Unreach:                24 / 153 / 83
Hosts Flapping:                       0
Hosts In Downtime:                    0
```

Linux System Utilities

The Linux Utility, top

Another assessment of performance is CPU Load. This metric is expressed in how many processes are in a runnable state averaged over 1 minute, 5 minute and 10 minute intervals. If this number is, say 4, that means that there were 4 runnable processes on average over the minute intervals. The following top output shows load averages of 0.58, 0.63, 0.64. These numbers state that about half of the time there were no processes in a runnable state.
Top's main strength is in showing you which processes are in what states. Which processes are taking the most resources? Top can show you. A typical scenario is where the Java listener that accepts feeds of data and posts them to the Collage DB, and the MySQL daemon, are the most active processes.
Top also gives you Memory and Swap metrics. These can also help pinpoint why a system is slowing. So, load numbers above 3, which means that, on average, there were over three runnable processes, with only one running and the others waiting to run, would indicate a slowing system.

The Linux Utility, uptime

Uptime is a smaller footprint tool that you can use to get the same load averages that top had.

```
# uptime
15:05:52 up 14 days, 22:55,  4 users,  load average: 0.46, 0.52, 0.52
```

The Linux Utility, vmstat

Vmstat has many options to it. Each option may output totally different types of metrics. One of the most useful is vmstat -s.

```
[root@lunias ~]# vmstat -s
      1034160  total memory
      1016344  used memory
       639180  active memory
       193732  inactive memory
        17816  free memory
       168028  buffer memory
       298856  swap cache
      2031608  total swap
       118044  used swap
      1913564  free swap
     32897831 non-nice user cpu ticks
         2581 nice user cpu ticks
      9633122 system cpu ticks
     79034644 idle cpu ticks
      8014841 IO-wait cpu ticks
       122537 IRQ cpu ticks
            0 softirq cpu ticks
    113646240 pages paged in
    972813361 pages paged out
     15889354 pages swapped in
    184057520 pages swapped out
   1347364576 interrupts
    959223271 CPU context switches
   1161043831 boot time
      7213271 forks
```

Many useful metrics are here: free memory, free swap and pages paged/swapped in/out. These metrics tell you whether you have enough memory.

Linux utility: sar

Sar is the system activity reporter. By interpreting the reports that sar produces, you can locate system bottlenecks and suggest some possible solutions to those annoying performance problems. The Linux kernel maintains internal counters that keep track of requests, completion times, I/O block counts, etc. From this and other information, sar calculates rates and ratios that give insight into where the bottlenecks are.
The key to understanding sar is that it reports on system activity over a period of time. You must take care to collect sar data at an appropriate time (not at lunch time or on weekends, for example).
A good sar tutorial can be found at http://perso.orange.fr/sebastien.godard/tutorial.html.

Other sar Utilities

- iostat
- sadc
- sa1
- sadf
- mpstat

Installing & Configuring sar (Sysstat)

- http://www.linuxfromscratch.org/blfs/view/svn/general/sysstat.html

RRD Performance Data Collection

GroundWork can collect performance data using an eventhandler. It runs after each check finishes, reading a DB Table and trying to log performance data. Performance data collection can contribute to a slow system.
The simplest way to gauge this is to count the RRD files in /usr/local/groundwork/rrd that have been modified that day.

```
# cd /usr/local/groundwork/rrd
# ls -chot *.rrd | egrep 'Oct 27' | wc -l
93
```

GroundWork Subsystems

There are various log files and other audit aids in GroundWork Monitor that can give you hints about how well the server is performing.

Process Performance Data Event Handler Log

For the purposes of this discussion, we will want to focus on the eventhandler logfile, /usr/local/groundwork/nagios/var/log/process_service_perfdata_file.log.

Output to this logfile is controlled by a debug_level variable, at the beginning of the perfdata properties file

/usr/local/groundwork/config/perfdata.properties

```
# Possible debug_level values:
# 0 = no info of any kind printed, except for startup/shutdown
#     messages and major errors
# 1 = print just error info and summary statistical data
# 2 = also print basic debug info
# 3 = print detailed debug info
debug_level = 1
```

First, this debug_level variable only needs to be set to a value above 1 when you are actively debugging something in the Performance data collection subsystem. Otherwise, it should be set to 0 or 1.

The contents of the log can be illuminating. This is a special-purpose eventhandler, one that runs after each service check that has performance data handling enabled - that's the default. Many checks, though, don't have a metric - such as any check that returns true or false. To tune the system, you should disable performance data collection for those services whose results aren't suited for such.

Also, just because you could plot the data on a graph doesn't mean that you should. Conservation of effort here will pay off. If no one is asking for the data, and you aren't particularly interested either, then turn it off.

To turn off performance data handling on a particular host or service, go into the Service record in the Monarch 'Configuration' Tab and, in the 'Service Detail' Tab, uncheck the 'Process Perf Data' property. In the 'Host Detail' Tab it is called the 'Process performance data' property.

Executing the Process-Perf-Data eventhandler only for services that actually are set up for performance data collection and not others, and turning off the logging function when not actively tuning the system, will save computing resources.

GroundWork Diagnostics Tool

GroundWork Customer Support has created a tool that methodically collects the above information and more. You could run the GroundWork Diagnostics Tool and send the information to Support or use it in your own analysis. See the kb article for more information: Running the gwdiags diagnostic tool

Conclusions

Having assessed your system and tuned it as well as possible, you might find that you still have a performance issue. Contact Support, or your GroundWork Account Representative to explore other options.