Purpose

This document describes how to maximize the performance of your Nagios XI server in a non-distributed environment in order to increase the overall efficiency of the monitoring server. This document will cover the primary causes of performance degradation and look at some possible solutions to maximize the use of your Nagios XI server. Specifically it will discuss maximizing active checks on a single Nagios XI server.

Target Audience

This document is intended for use by Nagios XI Administrators who wish to increase the efficiency of their Nagios XI implementation.

Summary

Nagios XI has several key factors that determine system performance. An administrator should have a solid understanding as to what is affecting their system's performance. There can be a lot of factors to consider because of the flexibility of Nagios XI. This document will cover the primary causes of performance degradation and look at some possible solutions to maximize the use of your Nagios XI server without having to resort to advanced system configurations.

Key Factors That Affect Nagios XI's Performance:

- Check Load = Total number of hosts + services.
- Check Interval = How often the checks are running
- Host and Service Latencies = The amount of time a check lags behind it's scheduled check time
- AJAX updates in Nagios XI's web interface
- Database Activity
- Subsystem Processes
- Disk I\O and Using RAM Disks

This document is based largely on a set of benchmark tests run on a 3gz Single Core system with 3.5GB of RAM. Most of the examples in this document will be based on that test environment.

Maximizing Performance: The Basic Formula

The check load of your monitoring environment is made up from the total number of active checks being run on your Nagios XI server (hosts + services). This number can range from hundreds to thousands depending on your monitoring environment. The bottom line on active checks comes down to: *How many checks can your system handle per second?*

This number cannot be determined without considering your average check interval for all of your hosts and services. In Nagios XI, the default check interval is typically 5 minutes. If you create thousands of checks in XI without ever adjusting this number on any of your checks, then all of your checks will be happening within a 5 minute window. This can quickly push your CPU very hard and limit the max number of checks that your machine can handle.

Additional things that can affect performance of your monitoring solution include:

- Disk I/O subsystem performance
- The use of a ram disk for decreasing I/O for check results and performance graphs
- The placement of database servers
- The use of CPU intensive checks (SNMP, check_esx3.pl, and custom plugins)
- The use of passive vs. active checks
- How many users are accessing the monitoring system

Benchmark Tests

In our benchmark example, we ran a test with roughly 5200 checks (mostly PING, HTTP, DNS Res, and DNS IP) within a 10 minute interval on our single core machine.

Check Load = 5220

Check Interval = 10 minutes

5220 / 10 / 60 seconds = **8.6 checks per second**

Average CPU Load = **8.0 – 12.0**

Host/Service Latencies = **< 2.0 seconds**

With these settings the system was still usable, but it was sluggish and not optimal for regular use. The host and service latencies were generally under 2 seconds, which was high and indicated that the system could quickly become overtaxed with the average load running that high.

Reaper Settings

After some experimentation and studying up on performance tuning tips from large environment users of Nagios Core, we came across two settings that allowed us push the single core system much harder, while maintaining a much lower CPU load and check latency.

In the main Nagios Core config file `/usr/local/nagios/etc/nagios.cfg` there are two "reaper" settings that made a substantial difference when running a high volume of checks in a small time period. The "reaper" in the Nagios Core monitoring engine is a function that processes or "reaps" check results as they come in. This process occurs at specified intervals, and as the volume of checks increases, this process may need to happen more frequently in order to keep up with the amount of checks coming in.

Default reaper settings:

```
check_result_reaper_frequency=10
max_check_result_reaper_time=30
```

Modified reaper setting for a high check volume:

```
check_result_reaper_frequency=3
max_check_result_reaper_time=10
```

Once we modified the reaper setting we were able to bring the average check interval back down to 5 minutes with these results.

Monitoring Engine Performance

Metric	Value
Host Check Latency	
Min	0.00 sec
Max	0.00 sec
Avg	0.00 sec
Host Check Execution Time	
Min	0.00 sec
Max	10.00 sec
Avg	1.04 sec
Service Check Latency	
Min	0.00 sec
Max	0.00 sec
Avg	0.00 sec

Check Load = 5220

Check Interval = 5 minutes

5220 / 5 / 60 seconds = **17.4 checks per second**

Average CPU Load = **5.0 - 9.0**

Host/Service Latencies = **~ 0.5 seconds**

Metric	Value	
Load		
1-min	1.85	
5-min	1.07	
15-min	1.00	
CPU Stats		
User	18.40%	
Nice	0.00%	
System	2.40%	
I/O Wait	0.00%	
Steal	0.00%	
Idle	79.20%	
Memory		
Total	1877 MB	
Used	1,546 MB	
Free	330 MB	
Shared	0 MB	
Buffers	179 MB	
Cached	675 MB	

Even with the check volume doubled, the page load times improved, and although the CPU is being pushed, the XI interface can be actively used without the check latencies increasing and the check schedule from falling behind.

It should be noted that as the total number of checks increases past 5000, the max possible checks per second will decrease due to the Core monitoring engine having to work harder to maintain the check schedule. At 10000 checks, the maximum checks per second with any check interval on the benchmark test was around 13 checks per second.

Nagios XI Performance Settings

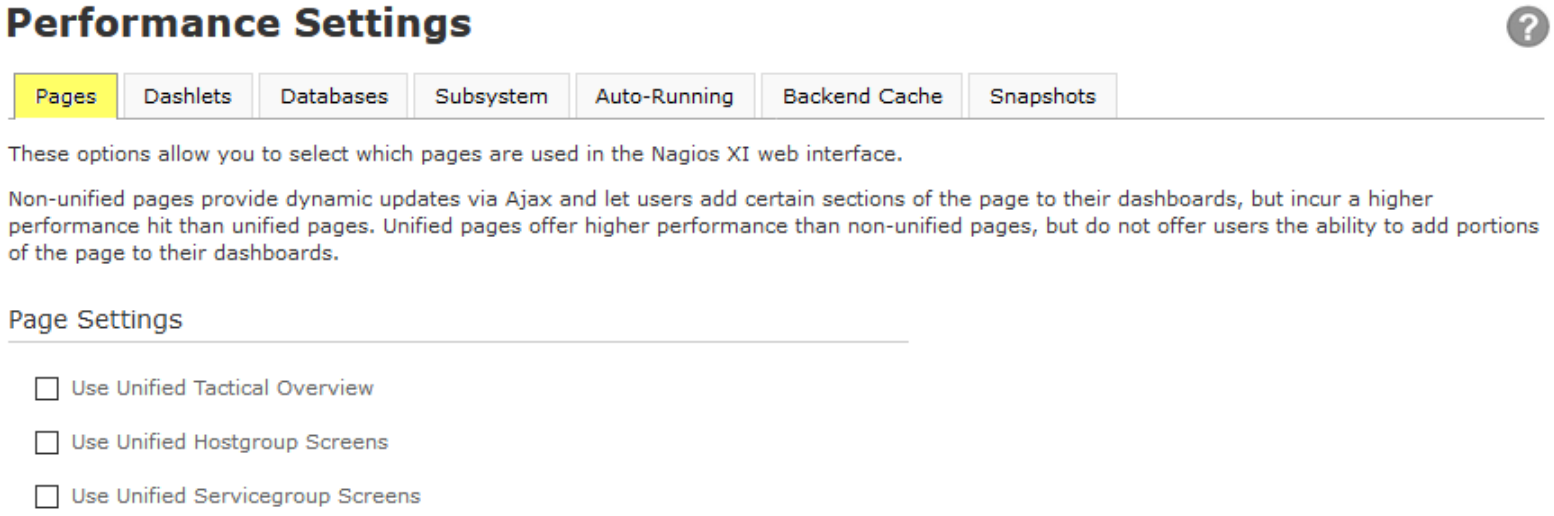
One of the key differences between the Nagios XI interface and the Nagios Core CGIs is the AJAX updates that are performed in Nagios XI. Nagios XI grabs XML data from the backend via AJAX requests in order to keep the data in your browser window up to date without have to reload the page.

The challenge this creates for system performance is that opening multiple browser windows to Nagios XI will have a dramatic increase in your system load. For every browser opened to Nagios XI, an additional set of AJAX calls are being made to the server for fresh data.

In order to reduce the load related to AJAX calls, Nagios XI allows you to use non-AJAX pages for various screens in its web UI. These settings can be adjusted in the latest version of Nagios XI by accessing the **Admin > System Config > Performance Settings** page. If you're running a high number of checks (5000+) or your CPU load is consistently high, there are a few adjustments you can make that may improve performance. These are described below in more detail.

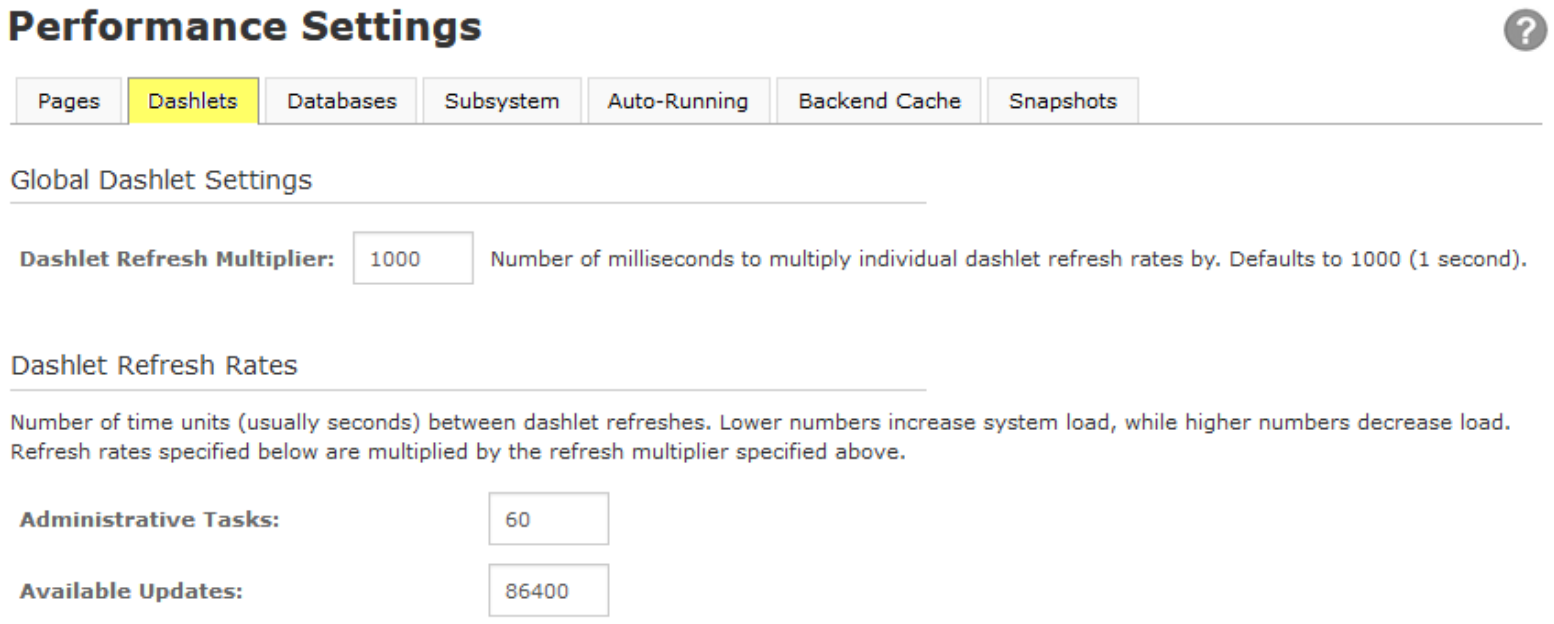
Performance Settings: Pages

Use unified dashboards to decrease the number of AJAX calls to the server. The unified tactical overview makes only one call to the server every 90 seconds and will noticeable decrease your CPU load.



Performance Settings: Dashlets

To globally adjust the refresh rate of all of your dashlets, increase the *Dashlet Refresh Multiplier*. The default value for this is 1000ms (1 second). You can also increase the refresh time for specific dashlets as needed in order to fine tune the performance to meet the needs of your monitoring environment *(not all shown below)*.



Performance Settings: Databases

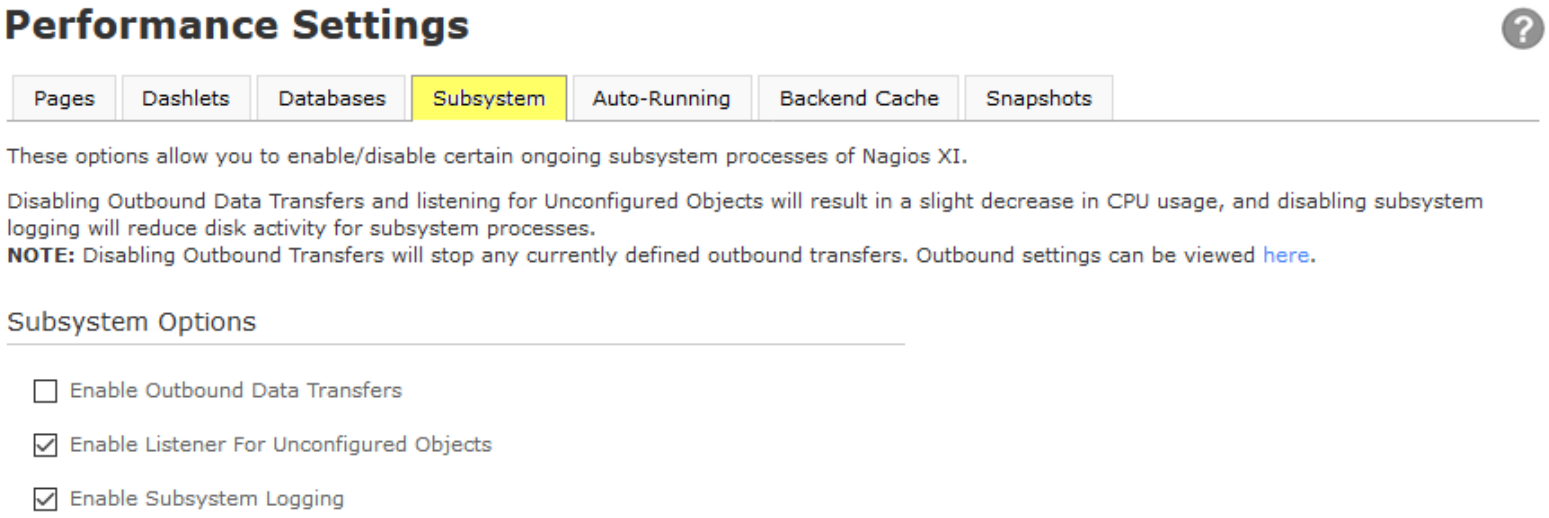
Database optimization for Nagios XI is a larger topic, and is covered in more detail in the following document:

Nagios XI Database Optimization

Performance Settings: Subsystem

Nagios XI relies on several subsystem processes that run on a continual basis. Changing these settings can result in a slight decrease in CPU usage. These settings have a more pronounced effect on larger systems.

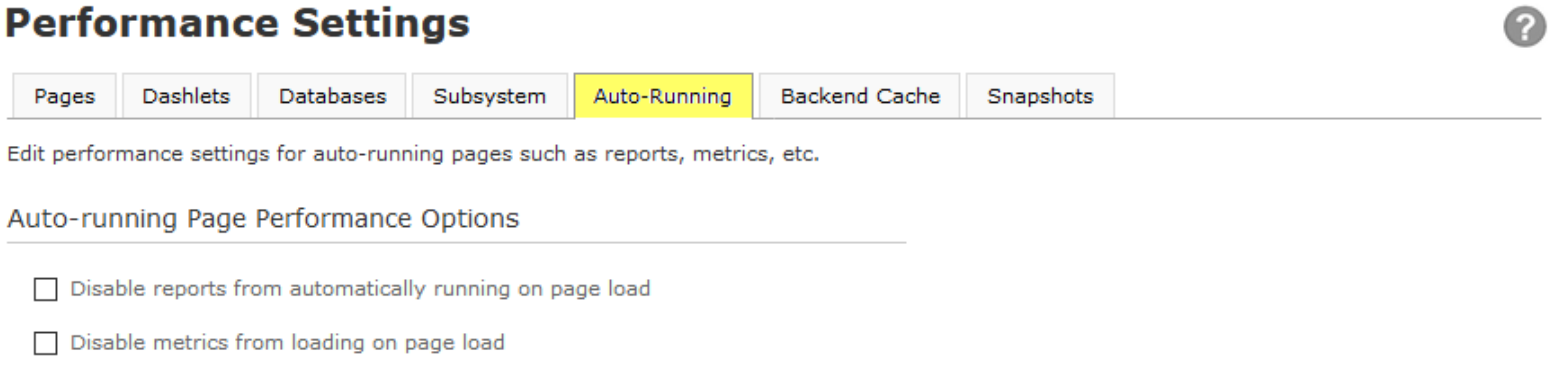
Disabling Outbound Data Transfers and listening for Unconfigured Objects will result in a slight decrease in CPU usage, and disabling subsystem logging will reduce Nagios XI's subsystem logging to a minimum, and will reduce disk activity and CPU for most subsystem processes. This can help reduce system usage during larger outages where many event handlers and notifications are being issued.



Performance Settings: Auto-Running

By default, when you visit a Report or view the Metrics page (via **Home > Details > Metrics**), the report or metric will automatically be generated. This can cause some considerable delays on larger Nagios XI systems.

These options all you to disable the automatic running of the reports and metrics. If the appropriate box is checked, the report or metric will not be run until the **Update** button is pressed. This gives you time to select which hosts, services, or groups you wish to include before the action is taken.

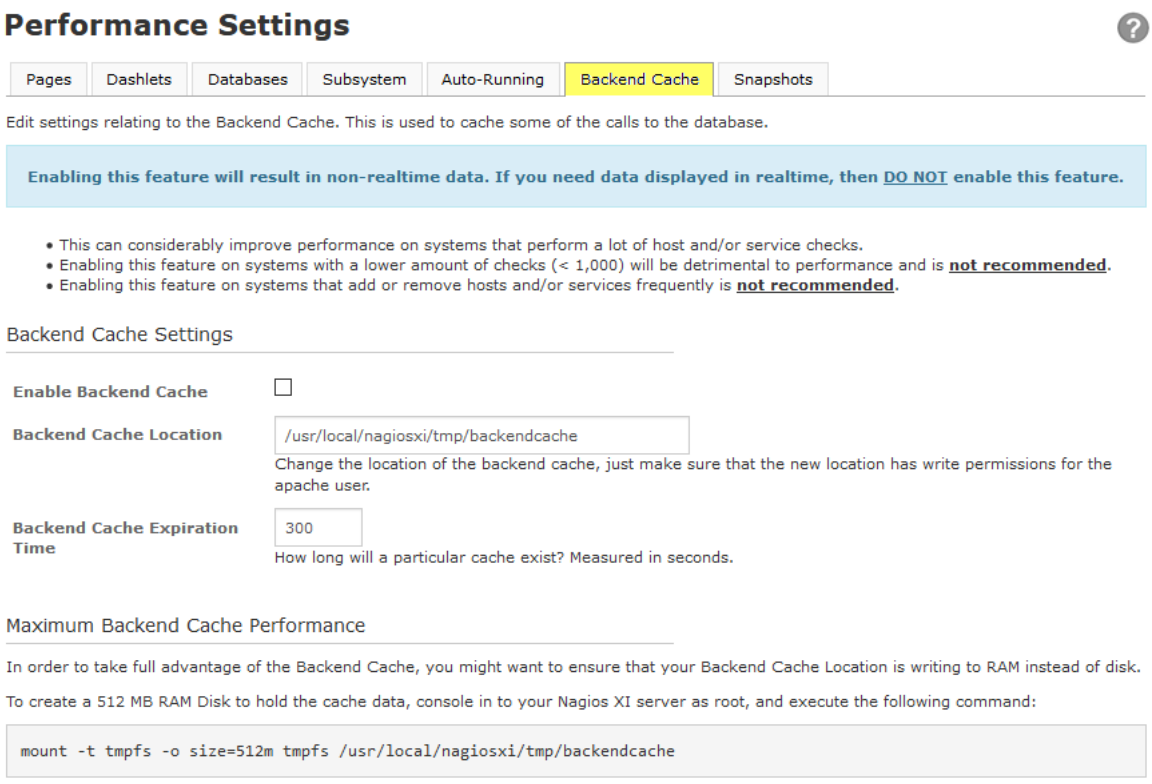


Performance Settings: Backend Cache (Nagios XI 5.3.0 or newer)

This is used to cache some of the calls to the database.

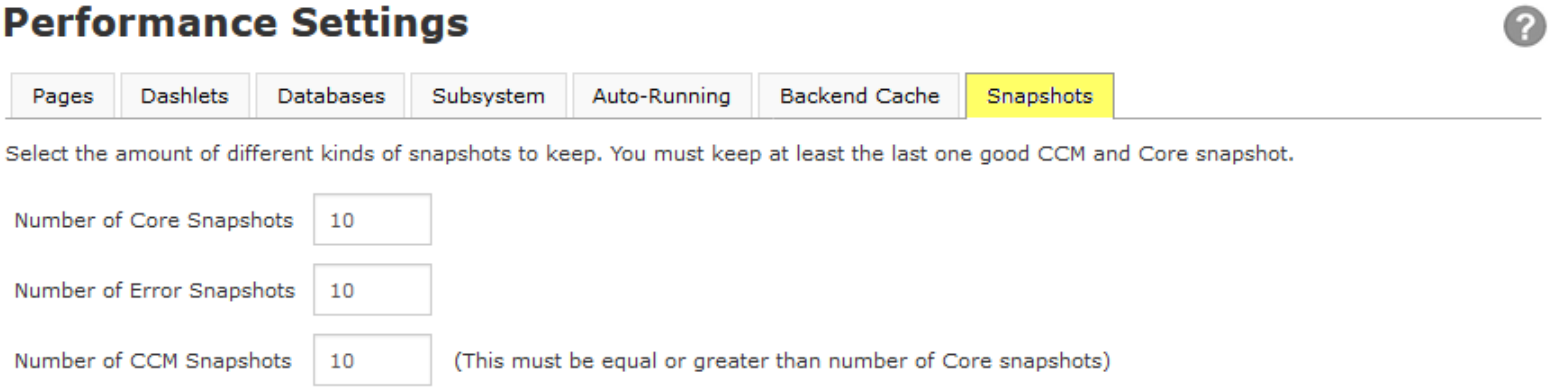
Enabling this feature will result in **non-realtime** data. If you need data displayed in realtime, then **DO NOT** enable this feature.

This can considerably improve performance on systems that perform a lot of host and/or service checks.



Performance Settings: Snapshots (Nagios XI 5.5.0 or newer)

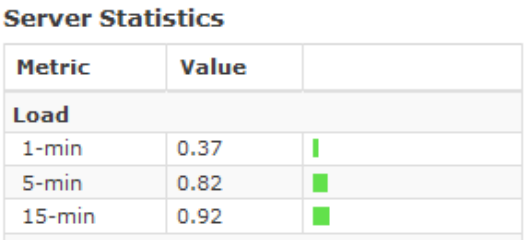
Core Config Manager (CCM) generates snapshots as part of the Apply Configuration process. The amount of snapshots you wish to retain can be tuned to your retention requirements.



Checking Performance

The following metrics can help you determine whether or not your system is able to keep up with the monitoring engine's check schedule:

- Host and service check latencies
- Monitoring engine event queue
- System load



These metrics can be found via **Admin > System Information > Monitoring Engine Status**.

Every CPU has a maximum effective number of checks per second that it will be able to handle. This number will vary based on the number of total checks being run, and what check plugins are being run. Compiled plugins are much more efficient than perl, python, and php (interpreted) plugins, and monitoring switch and router bandwidth also tends to use more CPU than most.

There are several dashlets included with Nagios XI that can be used to watch for overtaxing of your system.

See the examples to the right for other obvious signs of overtaxing your CPU. If your latencies are averaging over 10 seconds on a regular basis, then you either need to increase your average check interval, decrease the number of checks, or add more CPU power.

On overtaxed systems, the **Monitoring Engine Event Queue** will look extremely weighted on the left side.

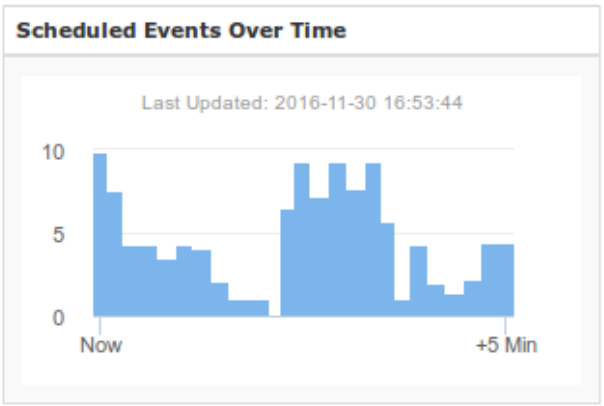
The images below provide an example:

- Left screenshot shows the monitoring event queue on a well-performing server
- Right screenshot shows an overtaxed system that will exhibit high check latencies

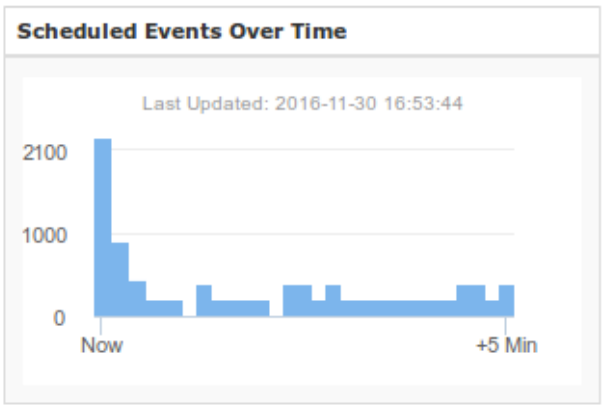
Monitoring Engine Performance

Metric	Value	
Host Check Latency		
Min	0.00 sec	
Max	0.00 sec	
Avg	0.00 sec	
Host Check Execution Time		
Min	0.00 sec	
Max	10.00 sec	
Avg	1.04 sec	
Service Check Latency		
Min	0.00 sec	
Max	0.00 sec	
Avg	0.00 sec	
Service Check Execution Time		
Min	0.00 sec	
Max	30.01 sec	
Avg	0.67 sec	

Monitoring Engine Event Queue



Monitoring Engine Event Queue



Monitoring Performance Using Nagiostats

The Nagiostats Wizard is a useful tool which will allow you to run checks against your internal Nagios performance and generate performance graphs for the actual program statistics. This can be useful for monitoring check latencies, execution times, and other performance analysis. The Nagiostats Wizard comes included with Nagios XI however if it cannot be found it can be downloaded from the Nagios Exchange:
https://exchange.nagios.org/directory/Addons/Configuration/Configuration-Wizards/Nagiostats-Wizard/details

Utilizing A RAM Disk

Although having enough CPU power on a Nagios server is important, the biggest hardware limitation to a Nagios system is disk I/O. A large Nagios installation creates an enormous amount of disk activity, and if the hard disk can't keep up with the constant traffic flow that needs to happen, even a large number of CPU's are simply going to wait for the disk in order to write new information to the disk. This can cause check latencies to soar even though the CPU usage appears within a safe range. Some users have solved this by using creative file mounts on separate partitions, or purchasing extremely fast disks for their servers.

The following documentation will cover how to configure Nagios XI with a RAM Disk:
Utilizing A RAM Disk In Nagios XI

Offloading MySQL To A Remote Server

The primary user of CPU on a Nagios XI system is ndoutils + mysql as a backend. The Nagios monitoring engine constantly writes fresh status information to the database backend so the information can be accessed in Nagios XI. On a large system, this accounts for the largest use of CPU on a Nagios XI install. The following document describes how to move the MySQL database to a second system in order to reduce CPU usage by up to 50%.
Offloading MySQL To Remote Server

Additional Options For Improving Performance

You can also look into distributed monitoring options and/or the use of passive checks to improve the performance of your system. These topics are covered in separate documents linked below.

Distributed Monitoring options for Nagios XI:

How To Implement A Distributed Nagios XI Environment

Passive Checks for Nagios XI:

How To Configure Inbound Checks With Nagios XI

How To Configure Passive Services In Nagios XI

Using NCSA With Nagios XI

NRDP Overview

Finishing Up

Managing your system performance in a Nagios XI environment is not a black and white formula, but a balancing act with the resources available on your server. Between the configuration changes described above (using a RAM disk, and offloading MySQL) a single Nagios XI server can be configured to handle well over 10,000 active checks. The key ideas to keep in mind are: How many checks do you need to run?, How often to you need to run them?, Can your hardware handle the check load it has been given?

This completes the documentation on maximizing performance in Nagios XI.
If you have additional questions or other support related questions, please visit us at our Nagios Support Forums:

https://support.nagios.com/forum

The Nagios Support Knowledgebase is also a great support resource:

https://support.nagios.com/kb