An AppDynamics Business White Paper
October 2013

Why Nagios and Server Monitoring Are Failing Modern Apps

Server monitoring is an important part of any data center monitoring architecture, but too often it becomes a crutch and a deterrent to successfully building out a holistic monitoring platform. Server status is only one indicator of application performance, so relying exclusively on server monitoring tools leaves organizations with large blind spots and unhappy end users. In this paper we will explore what server monitoring is and how it can (and should) fit into a larger application performance management platform.

What is Server Monitoring?

Server monitoring consists of monitoring operating system and associated hardware metrics for the servers that run your application. It's the view of the world from the perspective of the server, but never from inside the running processes. Basic server monitoring metrics include CPU sys time, CPU wait time, used memory, free memory, disk queue length, % disk used, network collisions, adapter transmit rate, etc. Server monitoring is used by every IT organization in some shape or form.

The 9s are an (Unintentional) Lie

IT organizations are usually held to a standard of three, four, or five "nines" of availability, referring to the number of nines in. The table below defines each of the "nines" and translates their meaning into acceptable minutes of server downtime per year.

THE NINES	YEARLY UPTIME (MINUTES)	MAX YEARLY DOWNTIME (MINUTES)
99.9% (3 nines)	525,074.5	525.5
99.99% (4 nines)	525,547.5	52.5
99.999% (5 nines)	525,594.8	5.2

Availability is usually measured at the server level by checking if the server is responding to requests. The problem with this method is that availability at the server level doesn't mean that the application is responsive or even available. If your servers are down, the application is down – but the opposite does not necessarily hold true. In order to truly address performance issues, IT needs to monitor more than just server availability – after all, what's the point of keeping an application up and running if it is so slow or error-prone that nobody will use it?

THE NINES	SERVER DOWNTIMECOST (DOLLARS)	APPLICATION DOWNTIME COST (DOLLARS)*
99.9%	???	5,255,000
99.99%	???	525,000
99.999%	???	52,000

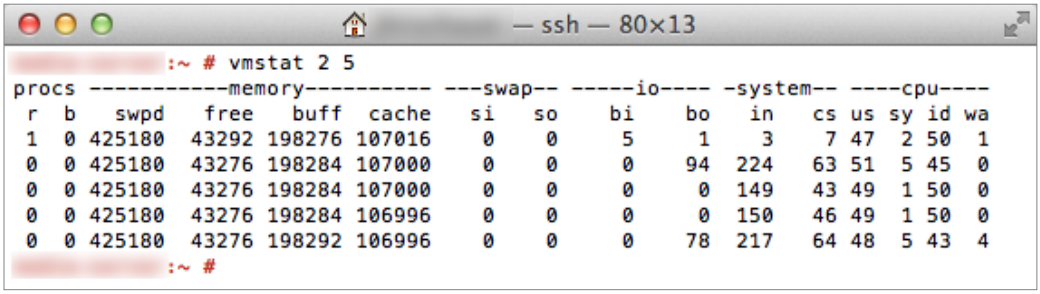
*Assuming the cost per minute of downtime is $10,000.

The chart above shows the costs associated with server and application downtime. Every company should track the cost of downtime for revenue-generating applications. No company can tell you the cost of server downtime without understanding whether or not the application(s) using those servers have been impacted.

What are Some Traditional Server Monitoring Tools?

Unix Systems Administrators use tools like sar, vmstat, nmon, top, topas, and netstat to monitor servers in real time. Windows Administrators use perfmon and WMI for real time monitoring. Other tools for alerting and storing historical metrics include BMC Patrol, HP OpenView, Miscrosoft SCOM, Nagios, Zenoss, Cacti, Zabbix, Ganglia, GroundWork, and Hyperic.

All of these tools are useful. *All of these tools also fall woefully short of achieving the goal of minimizing application downtime and maximizing application performance.*



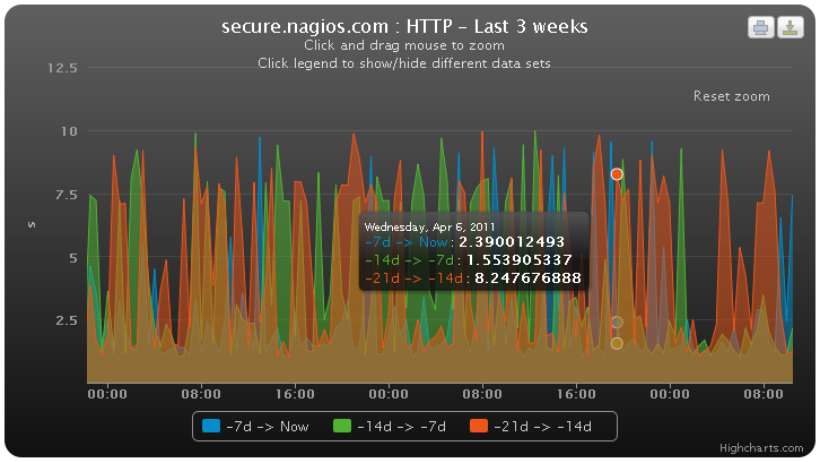
vmstat output from a Linux server. Is there any problem with the running application?

What's the Problem?

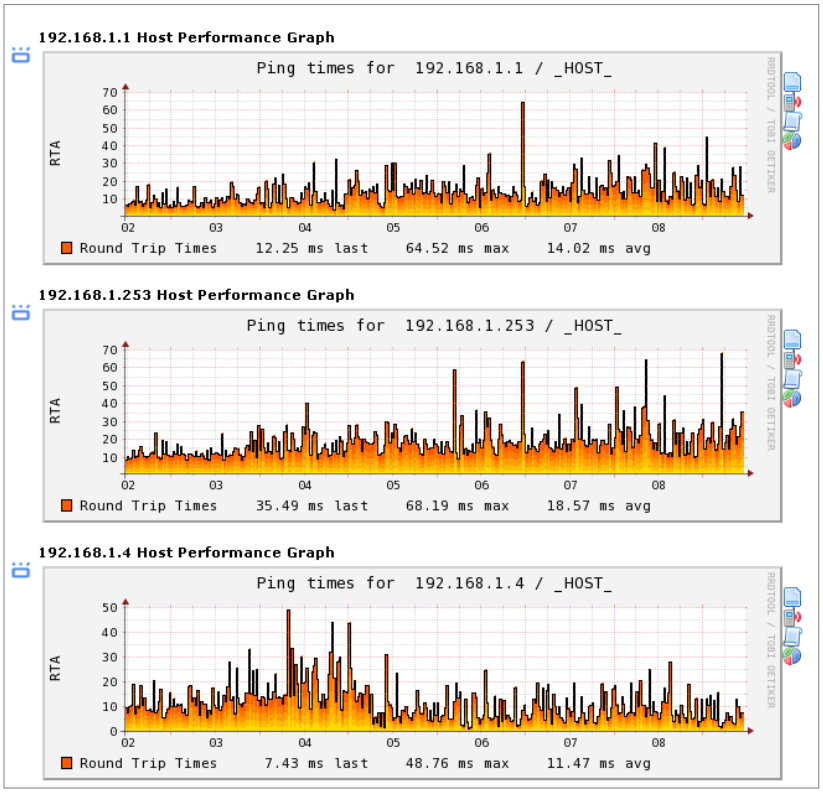
The problem is that none of these server monitoring tools are capable of knowing how your applications are performing. Some of them can probe your application to see if it is available or not but none can tell you why your application has ceased to function. No server monitoring tool can tell you any of the following:

- What is the response time of every request to my application?
- What components of my application are involved in any of my transactions and where is the slow down?
- How does the application code execute in the run time?
- What part of the application code is slow?
- What application functionality is used, how often, and how does it perform?
- What application functionality is throwing exceptions and what are they?
- Did a slow external service call impact my application response time and by how much?

Without answering those fundamental questions you don't stand a chance of restoring application service in minutes instead of hours or days.



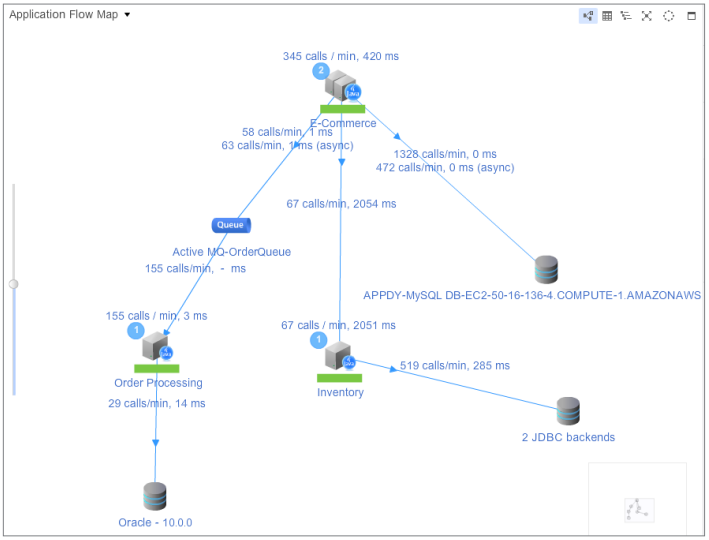
Nagios HTTP check response times. Is the application experiencing problems as a whole? Problems with individual functions? Are there application Errors?



Nagios server monitoring charts: What does this tell us about our application?

What's the Solution?

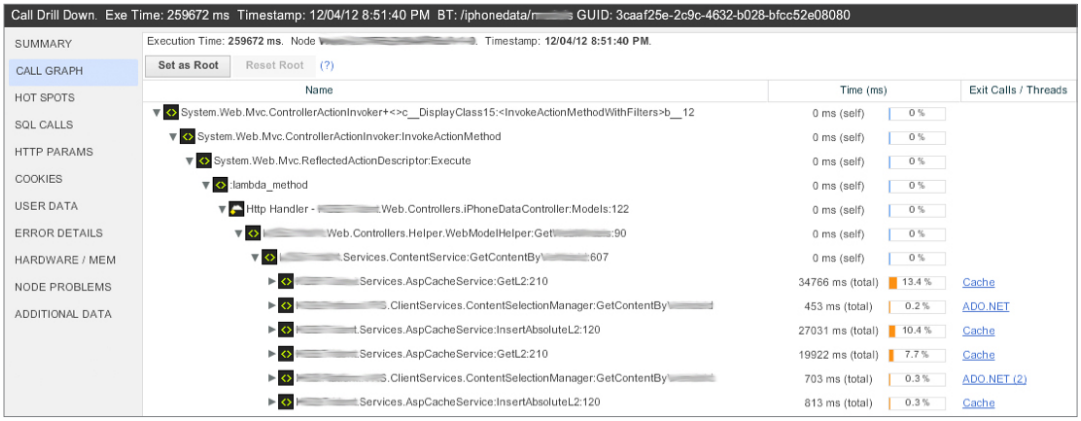
Many companies have turned to log monitoring and analytics as a partial solution to this problem. Log file monitoring is nice to have, but it can't answer many of the questions posed in the above section without a lot of customization. The best solution to the problem at hand is to use the latest generation of Application Performance Monitoring (APM) tools. APM tools understand the inner workings of your applications. They can see the code executing, the entry and exit calls to the application, the transactions flowing through and across multiple application components, exceptions and their associated impact, and much, much more.



Dynamic application flow map showing all application components.

Name	Health	Server Time (ms)	Calls	Calls / min	Errors	Error %	Slow Transactions	Very Slow Transactions	Stalled Transaction
Blog Homepage		220	21,121	5	0	0	292	602	-
Homepage		44	82,144	19	13	0	1,249	127	-
Solution		175	16,403	4	91	0.6	46	124	-
Blog Dashboard		708	312	< 1	0	0	3	14	0
Pro Trial		512	1,187	< 1	-	0	6	10	-
ROI		452	109	< 1	0	0	5	4	0
Blog RSS		102	216	< 1	0	0	4	2	0
node_add		270	10	< 1	0	0	0	1	0
Search		34	125	< 1	0	0	0	1	0
Marketo Check		0	2,609	1	0	0	0	0	0
Style		131	30	< 1	0	0	0	0	0

Business transactions automatically detected, tracked, and classified.



Call graph of a single business transaction with all methods, timing, and remote calls.

Making the transition from server monitoring to APM requires learning some new vocabulary. It's not difficult, just different from what most monitoring personnel are used to.

Business Transactions (BTs) – BTs represent unique functionality within an application. The easiest way to understand BTs is with an example. Most applications require a user to provide login credentials before they can use the rest of the application. When you want to access a web application you visit the login page, fill in the required fields and click the proper button to begin the authentication process. When you click the button there is typically a slew of downstream activity on the application server, database server, LDAP service, and potentially many other services that all work in coordination to service your request. All of the activity that is generated due to your login request would be considered a "Login" business transaction.

Application Flow Map – An application flow map (or topology diagram) is the visual representation of all of the tiers of your application and their dependent components or services. Application flow maps should be dynamically built and updated based on what the APM tool detects, not manually created.

Call Graph – Also referred to as a call stack. This is a tree-based list of application code that is executed on application servers in order to service a specified business transaction. Call graphs are used to determine what code is responsible for problems within an application.

Errors & Exceptions – Exceptions and errors represent failures in the application logic. However, not all exceptions or errors impact the business. An example of a common exception that is not an application problem is when a customer enters an invalid coupon code during checkout. This invalid code may cause an application exception but is not a problem that can be fixed. What's important is to be able to monitor and alert on the exceptions and errors that affect your business by interfering with the application's functionality.

What's the Impact?

Moving from a server-monitoring based approach to managing performance to an APM solution can add a lot of value for your organization. Here are just a few of the benefits of using APM:

- Reducing Mean Time to Resolution (MTTR) from hours/days to minutes.
- Faster development due to less time tracking down bugs.
- Fewer bugs released because they are easier to identify and remediate.
- Faster QA cycle due to rapid problem detection, isolation, and resolution.
- More stable production environment due to better development and QA.

Conclusion

Server monitoring is an important part of any IT organization's toolkit. Tools like Nagios provide real-time insight into the health of your datacenter, and server availability is an important metric to monitor. But if your business relies on applications, you can't stop there. Server health and availability is only part of the picture – in order to really understand how your end users experience your application, you need to monitor at the application level, too. The best solution for managing application performance should include both server monitoring and the application-level monitoring provided by application performance management (APM) products. With both of these tools in hand you can be more proactive about application performance and ultimately reduce the impact of performance issues on your end users.

About AppDynamics

AppDynamics is the next-generation application performance management solution that simplifies the management of complex, business-critical apps. No one can stand slow applications—not IT Ops and Dev teams, not the CIO, and definitely not end users. With AppDynamics, no one has to tolerate slow performing apps ever again.
Visit us at www.appdynamics.com.

AppDynamics

AppDynamics, Inc. 303 Second Street, Suite 450 North Tower, San Francisco, CA 94107 www.appdynamics.com