;login:

THE MAGAZINE OF USENIX & SAGE
August 2002 volume 27 • number 4

inside:

USENIX & SAGE

The Advanced Computing Systems Association &
The System Administrators Guild

monitoring strategies for high-availability web apps

by Neil Ashizawa

Neil Ashizawa is an SiteScope Product Manager responsible for System Monitoring solutions. Neil has been with Mercury Interactive since 1996. During his six years, Neil has held positions such as a Technical Consultant,a Technical Courseware Developer, and as a Manager of Program Development.

neila@merc-int.com

With the rapid proliferation and popularity of Web interfaces, companies everywhere have begun exposing internal data and methods to the network as never before. Sometimes, this is simply done as an in-house intranet, with a Web interface thrown on top simply to make the users happy. In other cases, this is a full-blown customer-interaction system. While this is all well and good, mission-critical network-enabled applications are difficult to test and maintain, especially as the network scales.

In a network with thousands of users, key network applications – even internal ones – can reside across several servers, in multiple databases, or even in many geographical locations around the world. A Web administration team has three primary goals:

1. Maintain reliable and secure access for authorized users (and deny access to others)
2. Maintain users' ability to perform business-related activities (for instance, in an e-commerce site, it's essential to keep the purchasing functionality working)
3. Provide users with a responsive, efficient, and effective interface

How to Monitor System Availability

Unless your site is static, not a source of revenue, or does not offer any value to your customers, you need to monitor it. In the Web world, "monitor" is a fairly nebulous term, basically including all the methods an administrator has at his disposal to verify that the site is functioning as expected.

Monitoring system availability might be one of multiple monitoring objectives for which an operations team might be responsible. Other objectives might include system-level monitoring for capacity planning and trending or application-level monitoring for diagnostics and optimization. This paper primarily focuses on availability monitoring through the use of network service monitors. These range from simple tools – such as ping, traceroute, and others that are included by default with most UNIX-based systems – to complex enterprise solutions that can perform in-depth, cross-platform monitoring.

However, just knowing whether or not all the servers involved are running is often not nearly good enough. In many cases, potential problems need to be highlighted as quickly as possible – preferably *before* the failure affects the users. For this, you need to track both key statistics on individual servers and the health of your entire network.

As an example, consider the network of XYZ Inc. The two sections of this network are:

- The external network that provides e-commerce, Customer Relationship Management (CRM) support, and knowledge-based resources for customers
- The internal network that provides interfaces to these same systems for employees

Both systems provide mission-critical services and are thus built with high-availability, fail-over-capable, hot-swappable equipment. A highly trained IT staff is on-site or on-call 24 hours a day. Everything is working perfectly… right?

Picture this: an executive, working late due to an upcoming conference call to a potential customer on the other side of the globe, discovers that the internal CRM systems are responding very sluggishly. She considers contacting IT, but the only technical person on site is on a different floor, and she doesn't have time to locate him. She figures that the system is only slow, not completely broken, and that IT probably already knows about this problem.

In truth, the primary database server for the CRM application has crashed. A backup server is now carrying the entire load, and its performance is not quite keeping up. The backup server itself has a problem: one of its cooling fans has dust in the housing and is no longer moving much air. The backup's internal monitoring software notified the IT manager of the problem right as he was going home at 5:45 p.m. He figured that the backup could run on its other fans for the rest of the night – he would have someone fix the problem in the morning. Besides, this is only the backup we're talking about.

When the executive talks with the potential customer, she finds that they're unable to access the network at all. She completes their order from her end, commenting on the problems in the system. She promises to have someone look into it. After several minutes, she finds the IT guy on a coffee break and tells him about the problem. However, without knowing which server or servers are having problems, he has to check each one individually.

When another important overseas customer attempts to connect to the external network at 3:15 a.m., they cannot retrieve any data, either. The backup database server's second fan just failed, and the system overheated and shut down as a safety measure designed-in from the start. The customer sends an email to the administrator's personal address. Unfortunately, the administrator has been out sick for several days, and his email is piling up unread.

The IT person finds the crashed primary server and restarts it. Thinking the problem is solved, he doesn't check the backup server. With the primary running, the network is once again functioning – albeit a bit slowly. He sends an email to the administrator.

Do you see the problem? Several hours of costly downtime could have been avoided if the staff had been fully aware of the status of the key servers and if the IT manager had been alerted as soon as the problem escalated. When the primary server crashed, monitoring the stats of the backup became even more crucial, since it became a single point of failure.

You not only want to know that your server is serving *something*, you want to make sure it's serving the *right thing*.

WHAT IT NEEDS TO KNOW ABOUT SYSTEM AVAILABILITY

To determine the comprehensive status of a large network application, you basically need to watch three key metrics. First, is the server running and accessible? System crashes, hardware failures, and extended power interruptions do happen – as an administrator, you need to know about these as quickly as possible. Even if your mission-critical servers have fail-over redundancies in place, you still need to know immediately when a server goes down. Accessibility is a closely related factor: it doesn't matter that your server is running smoothly if it's unreachable from the rest of the network.

Second, you need to know if the servers are responding correctly. Even if the server is up, its data may be corrupted; in the case of a Web server, files could be out of sync or links broken, or its pages could have been defaced by hackers. You not only want to know that your server is serving *something*, you want to make sure it's serving the *right thing*. This same point goes for database, application, and other back-end servers – just because the database server responds to SQL requests doesn't mean that it's responding with the correct data.

Ping monitoring is the most basic way to keep track of your servers' status.

Third, you need to watch the server's response times. If your server responds correctly but takes several seconds longer than normal, it may mean that there are more serious problems waiting in the wings. You might need another server, or you might need to adjust what's running on the server. For diagnostic and trending purposes, it is best to watch metrics such as CPU utilization and memory and hard disk usage directly, so you can either isolate the root cause of performance degradation or even anticipate potential failures and correct the problem before it occurs.

Basic Monitoring

Ping monitoring is the most basic way to keep track of your servers' status. Sending an ICMP ECHO_REQUEST packet causes the server to reply with an ECHO_RESPONSE, indicating that it is functioning.

Actually, this simply indicates that its hardware is active and that the network layer of the TCP/IP stack is correctly processing ICMP packets. It does not mean that the server is fully functional or that it will produce the correct response to other types of requests.

Even so, ping monitoring is an important part of an overall monitoring strategy. Besides indicating that the server is capable of a response, pinging a computer can also reveal information about network traffic patterns. If it takes the server longer to respond than usual, it may mean that it is deluged with packets or that heavy traffic across the intervening network link is interfering with the transmission of ICMP packets.

On a UNIX system, it is a trivial matter to set up a cron job to ping key servers every hour or so and then email the administrator if the ping fails or if the response takes longer than expected. One way of doing this is shown in code sample 1 on page 72. However, coordinating dozens of ping scripts into usable data can be a very complex project.

Keep in mind, too, that this script only confirms that the server is accessible from whatever machine you use for monitoring. Unless this unit is outside your firewall – and preferably on the other side of the Internet – you might not be aware of problems caused by external network issues. Admittedly, such issues are often beyond your control, but having remote testing centers, perhaps outsourcing them, can be an important part of your monitoring strategy. Of course, email notification might be problematic from an external monitor if your network is down. Some sites go so far as cellular telephone modem communications for extremely critical notifications.

URL AND CONTENT MONITORING

On the other hand, just using ping is clearly not enough for the vast majority of network situations, particularly the large, mission-critical applications discussed at the outset. The second key metric then comes into play: we want to make sure that the server is responding correctly.

In the case of a Web server, the simplest way to do this is to establish an HTTP session, request key pages, and compare the results to an expected norm. For instance, you could use wget and grep in a shell script to confirm that a supplied regular expression occurs in the server's output as expected. As an example of this, see code sample 2 on page 73.

The example script is somewhat limited. For instance, you might need to track how the server responds to different types of browsers. The script in code sample 2 could be modified to change the USER_AGENT string sent by the client, so that you can retrieve pages separately for the major browsers.

An important point to remember, though, is that it is fairly trivial to submit a request for a certain document from a Web server, but in most cases, watching static pages on a server is not nearly as important as monitoring dynamically created content. Often we need to be able to see how the server responds to a submitted form, and this requires supplying CGI POST data and then observing how the server responds to given data. For an effective test, it might be necessary to try a handful of different data types or sizes.

A shell script implementation of this level of functionality is possible, but this type of program should ideally be written in a friendlier language. Additionally, the complexity of that kind of solution is far beyond the scope of this paper.

For more advanced monitoring, it is important to keep track of resource usage inside the server.

ADVANCED MONITORING

So far, we have discussed monitoring techniques that basically examine how the server is responding from the user's point of view. To gain insight into why the server responds the way it does – arguably the information most needed to solve a problem – we need to probe deeper, digging out information from the nether layers of your application.

A simple way to keep an eye on your server's performance is to measure response time and notify the administrator if response latency spikes. If the server suddenly takes substantially longer to perform a given task, your users would probably be able to see this same delay.

Another important way to watch the innards of your application is by monitoring key values in your database. By checking certain totals, important metrics, or even test values periodically, you can make sure that your application is interacting with data as intended. In fact, a comprehensive monitor could retrieve information from the database and then perform some quick calculations to ensure that the values are correct.

Implementing this kind of solution as a shell script is possible, but it would be wiser to work in a language more suited for database interaction and data handling – for quick-and-dirty scripting of this kind of system, Perl is often a good choice. The key would be to stage regular SQL queries and then compare the returned data to expected values or expressions.

For more advanced monitoring, it is important to keep track of resource usage inside the server. On most UNIX systems, you can use rsh or SSH to run standard system commands, such as df, free, and ps, or to retrieve kernel statistics from the /proc pseudotree.

THE PROBLEM WITH A SHELL SCRIPT SOLUTION

Although a shell script like those in the code samples is a good quick-and-dirty solution to this kind of problem, it has a number of significant limitations.

Scalability: A multitude of shell scripts may get the job done, but it can be a pain to track down the various options in dozens of configuration text files. A company with a large, complex Web site and IT department, or with an organizational structure that calls for multiple notifications when a system enters a failure state, would need a solution far more robust than an odd assortment of scripts could provide.

Maintenance: If the data to be monitored changes frequently, the complexity of a do-it-yourself solution can scale quite rapidly. A commonly used solution is to use "metascripts" to administrate scores of worker scripts. This "solution" is kludgy at best – like a house of cards, it's fragile and requires continuous, careful maintenance.

Learning Curve: Additionally, without a unified, easy-to-use interface, nontechnical users cannot use the system to get information about system failures or to report new problems unless they can be trained to retrieve needed information. Generally, do-it-yourself solutions place the system in the hands of one highly skilled administrator, so if he or she left, even other technical staff members could require a great deal of time to figure out how the system works.

Accuracy: Finally, a homemade solution may overlook important statistics or monitoring techniques. Even if a certain solution meets your company's needs today, how much work will it take to retrofit it to interoperate with tomorrow's hardware? For enterprises in fast-moving vertical markets, maintaining a shell script solution could be a full-time job. Management now has to pick between monitoring accuracy and cost – and accuracy often gets the short end of the stick.

COMMERCIAL SOLUTIONS

In response to these issues, a number of companies offer commercial monitoring products.

Empirix offers a broad range of testing and monitoring solutions, including OneSight, e-Monitor, and the e-TEST suite. Combining measurements of user experience and server/network activity, Empirix provides monitoring solutions to track Web application performance from inside (OneSight) and outside (FarSight) the firewall. Empirix's e-Monitor can also perform end-to-end Web transaction monitoring evaluating the user experience of your site.

Keynote's performance management solutions are designed to enable you to take control of your Internet performance by effectively turning your data center into a fast, efficient triage environment. Keynote can monitor the overall end-to-end Web-based application for availability and can send configurable emails or pager alarms when TCP-enabled Internet connections, servers, and CGIs become inaccessible or return incorrect data.

NetIQ AppManager provides a centralized console to proactively manage virtually every component of a highly distributed Windows NT and Windows 2000 environment, from the physical hardware to business-critical server applications, such as Microsoft Exchange, SQL Server, Citrix MetaFrame, Lotus Domino, Oracle, SAP R/3, and Microsoft Internet Information Server. The latest version also supports monitoring certain UNIX systems.

BMC is currently rolling out version 7 of its PATROL monitoring software. This version has been extended to allow for more secure monitoring configurations and now supports new platforms, including SuSE Linux Enterprise Server (zSeries), Microsoft .NET, and Microsoft Windows XP. The installation routine has also been upgraded to make it a more straightforward process to set up or upgrade PATROL.

Freshwater Software's robust monitoring offering, SiteScope, is the only one from this list that not only can monitor but also runs natively on the three major server platforms: Windows NT, Sun Solaris UNIX, and Linux. SiteScope includes specialized monitoring tools for over 60 network protocols, system-level metrics, and enterprise software packages. A built-in Web interface allows users and managers to check the network's status and IT administrators to manage the entire system remotely. Versatile yet easy to use options allow for numerous configurations and a variety of alerting methods, including email, pager, SNMP, or customizable scripts. The optional SiteSeer module complements this system with outside-the-firewall monitoring from afar.

Don't delay in getting a solid monitoring system in place.

CHOOSING A MONITORING STRATEGY

Which monitoring solution is right for you? Only you can determine that. The key is to evaluate your network needs. If your network consists of only a handful of servers running just one or two Web-enabled applications with fairly simple interfaces, a shell script solution may work well for you.

In a highly homogeneous network, you'll want to find the solution that best fits your chosen platform. If, though, you plan to expand into other platforms in the near-to-mid future, or if your network is currently a best-of-breed composition of numerous products, you'll want to make sure that the monitoring solution you choose is flexible enough to deliver useful information about every server system you use.

You may also want to choose a solution that provides a wide breadth of monitoring types so that you can not only monitor the availability of your network components but monitor system utilization for capacity planning or application-specific monitors for diagnosing bottlenecks. To take it a step further, you may want a complete solution for all of your monitoring requirements, correlating all of the metrics together to gain a holistic view of the entire infrastructure.

Returning to our earlier example, XYZ Inc. will probably want a solution that includes both internal and external monitoring points and specific monitors for each of their mission-critical applications. The system ought to have customizable logic that can proactively respond to minor failures as well as signal a critical alert when several otherwise minor failures occur at the same time. Additionally, XYZ probably needs a system that can get the attention of its technical staff via several methods – emailing the manager, giving console alerts to the on-site staff, and paging the administrator when he's gone home.

Conclusion

Monitoring is a very important part of a company's information services, a key to good customer relations in the Internet-enabled world. Think through your options thoroughly, but don't delay in getting a solid monitoring system in place. If any of your IT systems even approximate "mission-critical" status, the cost of allowing them to fail unnoticed for hours is far greater than the deployment cost of any monitoring solution.

Good monitoring enables you to achieve the primary goals of any Web-enabled application: ensuring that users can reliably access needed functionality, keeping the doors open and the cash flowing in at your e-commerce storefront, and providing users with an enjoyable experience overall, free from crashes, hang-ups, or irritating sluggishness. If these are your goals, if your site contains dynamic content, if it is a revenue-generating project, if it offers value to your customers, or if it is mission-critical in any sense of the word, you need monitoring software. But beyond that, you need a monitoring strategy – a cohesive view of what your network is, what you want it to be, and how monitoring it will help you get there.

CODE SAMPLE 1: pingmonitor.sh

```
#!/bin/bash
#
# pingmonitor.sh — simple monitoring with ping
#

# verify that the config file exists and is readable
if [ -r /etc/pingmonitor.conf ]; then

for machine in `grep -v '^#' /etc/pingmonitor.conf`; do
      # adjust -c and -w parameters as necessary for your network
      if ping -c5 -w10 $machine
      then : # do nothing, the host is up
      else # ping failed

# modify the message below as necessary
/usr/lib/sendmail -t -F 'pingmonitor.sh' << ENDMAIL
From: pingmonitor <admin@domain.net>
To: administrator <admin@domain.net>
Subject: Ping failed: $machine

Ping failed for "$machine".
Please verify that it is functioning correctly.
ENDMAIL

      fi
done

else # no config file
echo Sorry, it appears that your /etc/pingmonitor.conf file is missing echo or\
unreadable. No operations were performed.

exit 1
fi
```

To enable this monitor, mark it executable and add it as a cron job. On a RedHat Linux system, for example:

```
chmod 755 pingmonitor.sh
cp pingmonitor.sh /etc/cron.hourly/
```

You will also need to create a file named /etc/pingmonitor.conf. Every line in this file must either be a comment (starting with the # character) or an IP number or qualified domain name for the script to ping.

If the script can't read from its config file, it will print an error message and return a nonzero exit code, causing cron to then email the owner of the cron job (usually root). This could easily be modified to send email to an outside address, if preferable.

CODE SAMPLE 2: contentmonitor.sh

```
#!/bin/bash
#
# contentmonitor.sh
#

# verify that the config file exists and is readable
if [ -r /etc/contentmonitor.conf ]; then

tempfile="/tmp/contentmonitor.out"
while [ -e $tempfile ]; do
    # make sure we have a unique tempfile name
    tempfile = "$tempfile.$$.$SECONDS"
done

for lineno in `gawk '/^[^#]/ { print FNR }' /etc/contentmonitor.conf`
do

urltest=`sed -n "$lineno p" /etc/contentmonitor.conf | cut -f1`
testpat=`sed -n "$lineno p" /etc/contentmonitor.conf | cut -f2`

    wget —output-document=$tempfile $urltest

    if grep $testpat $tempfile
    then : # do nothing, the pattern was found
    else # failed, content has changed

# modify the message below as necessary
/usr/lib/sendmail -t -F 'contentmonitor.sh' << ENDMAIL
From: contentmonitor <admin@domain.net>
To: administrator <admin@domain.net>
Subject: Content change: $urltest

Content monitor test could not find "$testpat"
in "$urltest".
Please verify that the server is functioning and that the document
is correct.
ENDMAIL

    fi

rm -f $tempfile
done

else # error: no config file
echo Sorry, it appears that your /etc/contentmonitor.conf file is
echo missing or unreadable. No operations were performed.

exit 1
fi
```

Again, you will have to chmod this file to be executable and add it as a cron job.

You will also need a configuration file in /etc/contentmonitor.conf. Comment lines (beginning with the # character) will be ignored. Every other line must begin with the URL to be monitored, followed by a tab, and then a word, phrase (in quotes), or UNIX regular expression for the monitor to verify it is present in the retrieved page.