;login:
THE MAGAZINE OF USENIX & SAGE
August 2001 • Volume 26 • Number 5

Special Focus Issue:Clustering

Guest Editor: Joseph L. Kaiser

inside:

USENIX & SAGE
The Advanced Computing Systems Association &
The System Administrators Guild

monitoring tools for larger sites

by Stephen Chan,
Chan PDSF lead. He has spent the last 10 years working either as a system engineer of as a UNIX SA.

Cary Whitney,

Whitney has worked on PDSF at LBNL since 1999, and has played a key role in PDSF's past and ongoing development.

Iwona Sakreja,

Sakrejda started in PDSF User Services in June 2000). Prior to that worked for ten years at LBNL in the Nuclear Science Division.

and Shane Canon

Canon is a system administrator at NERSC where he helps administer a large linux cluster used for computational computing.

sychan@lbl.gov
clwhitney@lbl.gov
isakrejda@lbl.gov
canon@nersc.gov

The Parallel Distributed Systems Facility (PDSF) is a "Cluster of Clusters" managed by the National Energy Research Scientific Computing Center (NERSC) personnel under the auspices of the High Energy and Nuclear Physics Computing Support (HENPC) Group.

Introduction

One of the primary responsibilities of system administrators is to ensure that systems are running and users don't experience any service interruptions. As an environment becomes larger, with more services and more dependencies, it becomes increasingly difficult to track the state of your site. For small sites, or for sites with very specific requirements, administrators often create custom scripts or other monitoring tools to watch their environment and report any problems. However, as sites become larger and more complex, or monitoring policies become more stringent, it makes sense to look for existing tools and build upon them. So long as the tool is stable and well matched for your environment, this is an efficient approach. With the proliferation of the Internet and the burgeoning open source movement, there are more tools than ever before to monitor your site. This article covers some of the most popular, and discusses their design and operation. Tools exhibit the different design decisions of their creators; administrators will have to decide for themselves whether the design suits their requirements.

Our environment is a Linux cluster with roughly 180 servers, used for scientific computing at the Lawrence Berkeley National Laboratory. While it is not as large as some server farms, it is nonetheless large enough so that a monitoring tool not designed for performance and scalability will begin to show its limitations. The server we use for monitoring is a dual Pentium Pro 200Mhz (256K cache) with 64MB of memory. Not an especially powerful machine, but it provides a good platform for testing the efficiency of monitoring packages.

The intent of this article is to present a beginning overview of monitoring and describe several packages that provide a good starting point for further investigation. For readers who have some experience with monitoring, the later section that reviews several packages may be useful, in case you are trying to expand or upgrade your monitoring system. We won't be able to go into much depth in this article, but we hope to give readers a useful comparison of the tools available.

What Is Monitoring?

There are two classes of monitoring tools that are covered in this article: event (fault) monitoring and performance monitoring. Typically both are necessary in a production site, but for this article, we will spend more time on event/fault monitoring. The event monitoring tools we'll discuss are Big Brother, Mon, Big Sister, and NetSaint. For performance monitoring, we will focus on MRTG, one of the most popular packages available for trending network performance. Inevitably, if we talk about MRTG, the topic of SNMP comes up; however, SNMP is a very broad topic, and we cannot hope to do more than provide a high-level overview in this article. It should be clear that we are only covering free, open source tools. There are numerous powerful (and expensive) commercial packages available, but many of the free packages are very useful and more than adequate for many sites.

Event monitoring is essentially watching out for certain interesting changes in the state of your systems. Each such change is an "event." Of course, the term "interesting" is intentionally ambiguous: for most sysadmins, a server experiencing a kernel panic and crashing is "interesting," but something as seemingly benign as the utilization of a finite

For system administrators, often the three most important metrics are availability, utilization, and throughput

resource going above a certain threshold may be interesting as well (even if nothing crashes). This is why we use the term "event monitoring" instead of merely "fault monitoring." Fault monitoring implies that something is broken, but we may be interested in an event, even if nothing is broken (because the event may indicate that something might break soon). Generally, if an interesting event occurs, we want some kind of response to be triggered – it can be as simple as sending a message to a pager or as complex as starting a script that performs diagnosis and possible recovery.

Performance monitoring involves tracking metrics related to how systems are performing. For system administrators, the three most important metrics are often availability, utilization, and throughput. Availability is a measure of the percentage of time that a system is up and available to users, while utilization measures what percentage of the total capacity is in use (for example, what percentage of time a CPU is non-idle). A metric related to utilization that is often used in network monitoring is throughput, or the amount of activity per unit of time (for example, the number of megabits per second flowing through a network switch port).

Performance and event monitoring can overlap because the underlying metrics being gathered are often the same. For performance monitoring, these metrics are processed to produce graphs or some kind of summary statistics. For event monitoring, changes in the metrics, the inability to collect the metrics, or the inability to connect to a service (a "service check") are the events being monitored. As a simple example, if we try to connect to a server and discover that it is not responding, to the event monitor, this event may trigger a page to the person who is on call. To the performance monitor, the fact that the server is down is a data point for calculation of overall availability.

An important protocol for both event and performance monitoring is SNMP, the Simple Network Management Protocol. SNMP is a UDP protocol based on the notion of reading and setting variables that are tied to the state of devices on a network. By reading the value of a variable via SNMP, you can discover information about the device. By setting the value of a variable via SNMP, you can alter the state of the device. Networking hardware typically has SNMP support built in, and SNMP is the main standard used for remotely administering networking hardware. Computers and other devices on the network typically support SNMP as well, but it often requires configuration.

On a computer, SNMP typically requires that an agent be installed and running. This agent handles SNMP requests and can be configured to send SNMP event notifications (SNMP traps) under some circumstances. An SNMP agent can be extremely useful for gathering metrics and remotely administering machines. However, this utility comes with the requirement to administer the agent; an unconfigured or poorly configured SNMP agent is a security nightmare.

Most of the event monitoring tools discussed have their own custom agents. SNMP agents are the de facto standard for performance monitoring, but they can also be used for event monitoring. The choice of whether to use an SNMP agent or the custom agent is up to the administrator. Event monitoring packages usually prefer to use their own agents for gathering metrics, and their default configuration usually doesn't support SNMP agents. However, the security and robustness of these agents can be a big unknown – while SNMP is a security hazard, it is at least an understood hazard. Typically, an SNMP agent is much more general than the custom monitoring agents, and with the appropriate investment of time, it is more powerful and flexible.

Event Monitoring Packages

There are four network monitoring packages that we will discuss: Big Brother, Big Sister, Mon, and NetSaint. Each one is a free, open source package that can be found on the Internet. For the most part, these packages are stable and can be used in a production environment to do monitoring. All of these packages can be extended with plug-ins or, if you are so inclined, by modifying the source code. In addition, they all have Web interfaces – one package, Mon, has a command-line interface as well.

All of these packages will perform basic monitoring, such as pinging hosts or checking if common services (HTTP, Telnet, etc.) are listening. All of the packages support monitoring using *polling* ("pulling" information from services being tested) and some of them support *pushing,* in which clients send information to the central monitoring package ("pushing" information from the clients being monitored).

Polling tends to concentrate all the work on the machine that is doing the polling. Consequently, the polling machine can become bogged down. The benefit is that you have only a single point of administration, simplifying management dramatically. With pushing, where clients send information to the monitoring host, more of the load is borne by clients, but this can require more administration. It also means that an agent has to be installed on each of the machines being monitored, and these machines must be able to initiate connections to the monitoring host. This can be a problem if there are trust issues – for example, if the monitoring host is inside a firewall, but the monitored host is outside the firewall. You typically don't want external hosts initiating connections through the firewall. This is an issue that an administrator needs to consider very carefully in light of the security policies for his or her site.

All of the packages described support external plug-ins. These are custom written programs that monitor services that the basic monitoring package doesn't support. For the most part, the plug-ins are external scripts with well-defined exit values and output that let the system know the state of the tested service. Writing a plug-in in C and then compiling to native code is generally fastest in terms of performance; however, coding something in Perl is usually the most convenient approach. Some of the packages have an embedded Perl interpreter to speed up Perl-based monitors. This is an important consideration if you have a lot to monitor and prefer to use Perl. It is also useful because SNMP offers many monitoring possibilities that may not be supported in one of the canned monitoring tools.

Plug-ins are also valuable for testing more complex applications. For end-to-end testing of an application, it may be necessary to engage in an extended transaction (for example, testing an e-commerce application on a Web server) by writing a custom client. This is an important consideration for monitoring more complex sites.

Big Brother

Big Brother is one of the most popular packages available. It is straightforward to install and configure, and has a large user base that has produced numerous plug-ins for monitoring services. Big Brother is a combination of shell scripts and compiled C programs that will gather information and generate reasonably photogenic Web pages that provide up-to-date status information. If something breaks, Big Brother has a highly configurable policy-based notification system that supports email, pagers, and SMS. Notifications can be acknowledged via the Web page or an email message.

Big Brother also has a script that generates availability statistics, which is nominally a part of performance monitoring. With the base installation, you can monitor the fol-

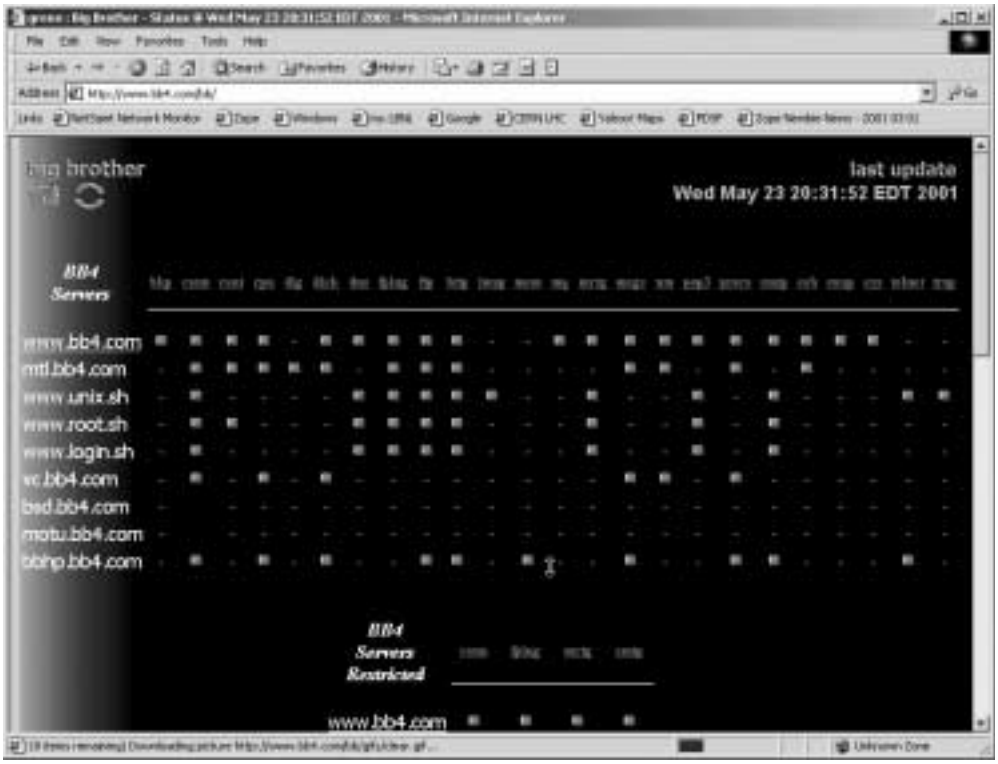

Figure 1: Demo Screen from Big Brother

lowing services: FTP, SMTP, POP3, Telnet, SSH, NNTP, DNS, HTTP, HTTPS. Note that monitoring of HTTP and HTTPS requires the Lynx browser. Plug-ins, in the form of external programs, are supported to monitor services that the base installation does not. Big Brother can monitor most services you can think of, and if a service isn't covered, you can easily write a plug-in using their interface.

If an agent is installed on a machine, it will push information about running processes, disk space, CPU utilization, and similar metrics to the Big Brother server. All this information can be tied to notifications as well, so if an important process dies and doesn't respawn, Big Brother will let you know. Even though Big Brother collects this information, it only stores enough to perform availability reporting and not utilization or performance tracking.

Big Brother does not have SNMP support built in, but it can be extended to support SNMP traps and SNMP polling via plug-ins.

Big Brother keeps its state information in a collection of text files. Since they are text files, they are relatively easy to parse; however, text is not the most efficient format for storing data, or for accessing data. If you keep historical data for a long period of time, the disk space usage really starts to add up.

Big Brother also has support for redundant Big Brother installations and some support for distributed monitoring. This is handy for remote sites and can also be used to scale up the capacity of Big Brother.

Our experience is that Big Brother is excellent for smaller sites, but it is missing some features necessary for monitoring larger sites. For larger sites, you need the ability to control the rate at which tests are being executed; if you have 700 different services being monitored, you don't want to run all 700 at the same time. By the same token, you would not want to run the tests one at a time, because the time to go through all 700 tests may far exceed the interval between tests. It is entirely possible that you will receive error notifications, not because something is down, but because you cannot effectively

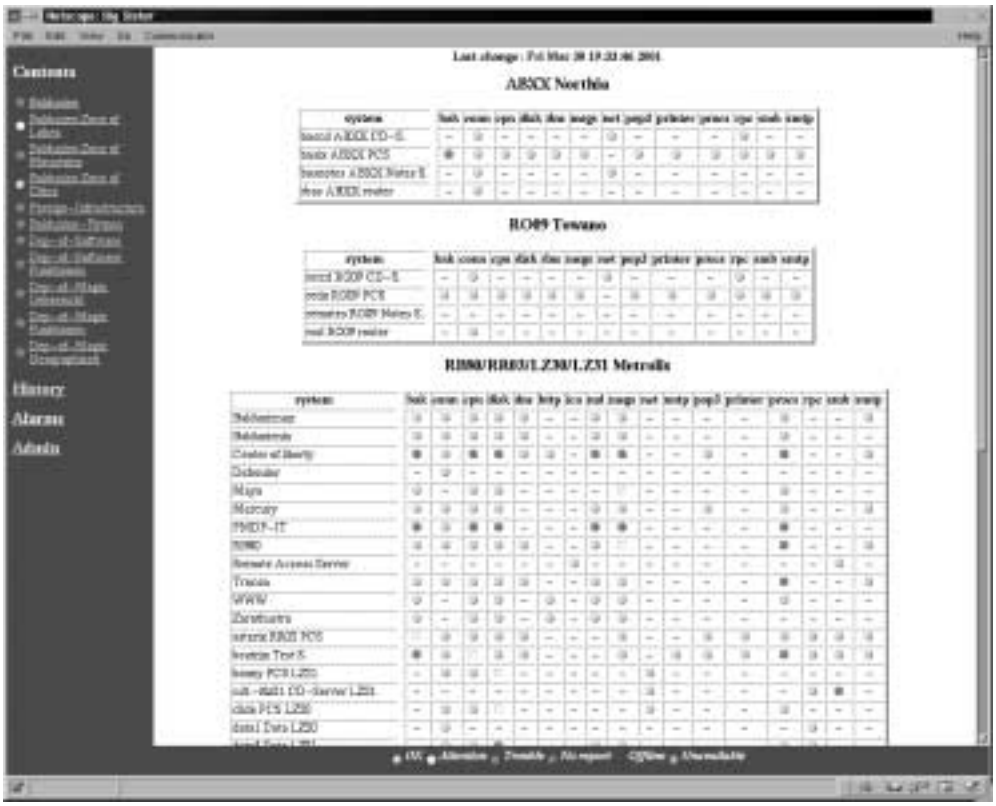

Figure 2: Demo Screen from Big Sister

perform all the tests within the allotted time. The portions of Big Brother that are in C run very quickly; however, the portions in Bourne shell are lacking in performance. Our experience is that Big Brother was not able to run all its tests quickly enough to avoid false errors showing up. It may be possible to avoid this by using a distributed monitoring approach; we chose to examine other tools, however, to see if they would work without requiring more hardware.

Big Brother's reporting tools are reasonable, but it would be better if the data that Big Brother collected were in a database with a standard interface, so that reporting tools could be leveraged to generate custom reports.

Big Sister

Big Sister is a clone of Big Brother. It is compatible with many of Big Brother's plug-ins and clients, and adds many new and useful features. One of the most basic differences is that Big Sister is implemented in Perl and uses the round-robin database tool (rrdtool) to store performance and utilization statistics for trending.

Big Sister's user interface is structurally similar to Big Brother, and it has most of the same features as described for Big Brother. Big Sister goes beyond Big Brother in the following areas:

Performance trending – Big Sister stores performance data in a database and generates graphs to describe performance and utilization. The database back end, rrdtool, is the same tool used for many network performance monitors and has C, Perl, and command-line interfaces for gathering information and generating graphs. If you need this feature, the rrdtool approach has benefits that will be discussed later.

SNMP support – Big Sister can use SNMP agents to gather statistics. However, it appears that SNMP information may not be used for trending. Big Brother supports SNMP as well, via plug-ins, so it is less well integrated.

Syslog parsing – Big Sister will examine the syslog file for errors or other lines matching configurable regular expressions. This is especially handy if you have a centralized loghost.

Tripwire – Tripwire is used to verify file permissions and checksums on important system files for security auditing. This can save you some time if security is a major concern.

Additional monitors – Big Sister has several other monitors built in, which are often covered by Big Brother plug-ins. For reference purposes, the additional monitors include: Oracle, RPC, SAR-based metrics, RADIUS, OpenView trap monitor. In addition, Big Sister comes with support for SNMP traps – event notifications that devices send over SNMP (instead of being polled, a device pushes an event to the server). The benefit of a built-in monitor is that they often have much better integration with the core package and can present information in more detail and many of the built-in Big Sister monitors provide a good level of detailed information.

Big Sister also uses rrdtool for storing and graphing performance metrics. rrdtool will be discussed more in the performance monitoring section; suffice it to say that this adds a lot of flexibility and power with respect to gathering and displaying metrics.

Generally speaking, Big Sister has more monitors in the base installation and better tools for reporting, especially generating graphical reports.

Big Sister does not really add any new functionality when it comes to support for larger sites. The fact that Big Sister is written in Perl is both a boon and a bane. Perl is excellent in terms of modifying and extending the software; however, Big Sister is relatively complex for a Perl script. Perl's garbage collector makes it easy to write code, but the reference-counting implementation tends to be "leaky" on long-running scripts that use lots of objects. In our tests, we found that the resident set size of the Big Sister application could get to 20MB within a few days. This was only aggravated by the fact that we had many machines and many services being monitored. This is not a criticism of Big Sister, but of the limitations of Perl for complex, long-running programs (especially if there are lots of anonymous objects being created, as typically occurs when using the OO extensions). If Big Sister were re-implemented in the latest version of Python, which has an improved garbage collector, or Ruby, with its Mark&Sweep garbage collector, this problem might be avoided. This problem can also be worked around with a nightly restart of the offending Perl scripts.

As memory usage goes up, performance on the monitoring host often starts to degrade. It appeared that Big Sister did not parallelize its tests. Combined with the slowdown from memory usage, it became impossible to complete all the monitors within the scheduled amount of time (15 minutes between tests), resulting in many systems appearing to go down and then coming back up a few minutes later.

Because of the lack of support for larger sites, and the problem we saw with core leaks, Big Sister, despite it's many attractive features, does not seem to be appropriate for a larger site.

Mon

Mon is a set of Perl scripts that, in terms of functionality, is one of the more basic tools covered. However, one of Mon's more interesting facets is that it has multiple interfaces into the system – supporting command-line, Web, and even two-way-pager interfaces. Mon is implemented in Perl, but apparently because the package does not implement all

the monitoring and trending features of Big Sister, it doesn't seem to suffer from the same garbage collection problems.

Mon can be described as a minimalist scheduling and notification framework for monitor programs. It schedules tests to be run and, based on the results, may send out notifications. It has a server that clients can connect to in order to query and update the state of the system, which then implements a Web, command-line, or other interface to the end user.

The following monitors are provided: asyncreboot (monitor host reboots via SNMP), dialin, DNS, fping, FreeSpace, FTP, hpnp, IMAP, LDAP, MySQL, netapp quota/FreeSpace, NNTP, ping, POP3, monitor processes via SNMP, rd (notifies if too many or too few files are in a directory), RPC, SMTP, TCP, Telnet, and network round-trip times. Like Big Sister, Mon supports SNMP traps.

Mon also has event handlers – which are programs or scripts that should be run when certain events are detected. For example, you could configure Mon to run a utility to restart a Web server on a remote machine if it notices that the server has gone down. This is a very useful feature for problems that are well understood and can be automatically resolved.

Mon doesn't have its own agents, preferring to leverage SNMP for many of these functions. In general, Mon is a straightforward monitoring tool. It does not provide a large feature list, but it is dependable, has a low administration overhead, and allows many forms of interaction. Its configuration is very clean and easy to read, more so than the other packages we examined. The Web interface also has useful controls for stopping and restarting the Mon server process. In terms of it's core monitoring and notification functionality, it is on par with most of the tools described, but it does not have a snappy Web interface, trending, or availability reports.

For large sites, Mon's scheduler has a useful feature: it can put a limit on how many tests are being run at any given time. We actually ran Mon for quite some time and it did a

Figure 3: Demo Screen from Mon

good job of watching the site without bogging down. If you need a basic level of monitoring that is dependable and straightforward to configure, but with very little reporting, Mon is a solid performer.

NetSaint http://netsaint.sourceforge.net/

NetSaint is a monitoring package that seems to have been designed for speed and scalability. The package is written in C, and virtually all of the monitors are coded in C as well. For folks who prefer Perl, the latest version includes an embedded Perl interpreter to eliminate the overhead of forking and execing a new Perl interpreter. The package also has one of the more visually appealing Web interfaces of the four we've examined. It should be stated that NetSaint is the package we are most familiar with because it seemed to fit our requirements well, and, as a result, we've invested more time in exploring it.

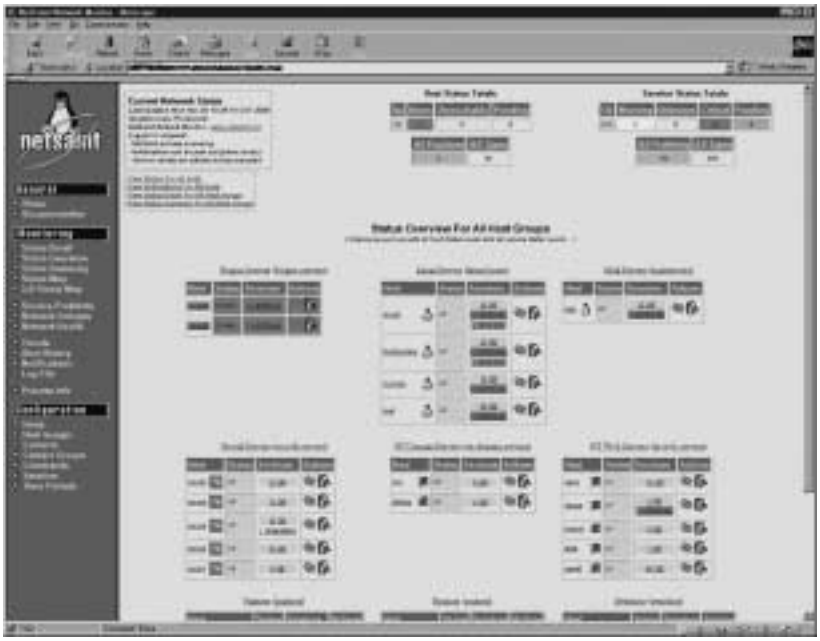

Figure 4: Demo NetSaint Screen

NetSaint has many useful features for supporting larger sites. It parallelizes service checks and provides directives for controlling the maximum number of service checks, as well as a method of interleaving the service checks so that checks are spread out uniformly. With these features proPerly enabled, NetSaint was able to easily complete 700 total service checks against ~180 hosts with extremely low load on the monitoring host (usually 90% idle).

Another feature that NetSaint supports is passive service checks. This allows service checks to be performed elsewhere and then sent to a NetSaint server for tracking and notifications. This distributes the load across multiple machines and can be very useful for large sites or for monitoring remote sites.

NetSaint also has many other useful features including exposing the number of and interval between retries before sending out an error notification. It will often be the case that there is a transient problem that causes a test to fail or time out. Under these circumstances, you would prefer that the monitoring package briefly wait before retrying the test to avoid a false alarm. In many packages you can modify the script or program that does the checking, but in NetSaint, it is parameterized in the configuration file. This was very handy in cutting down on false alarms against our more heavily loaded NFS servers.

NetSaint also generates reasonable availability summaries and trending graphs. However, NetSaint doesn't do any performance or utilization trending. It *does,* however, have an interface that allows performance data from tests to be passed to external trending packages.

NetSaint also has a very handy feature: when using the Web interface, it is possible to put notes on machines or add notes to services that have been marked as "down." This is an excellent feature for communicating between UNIX system administrators, operators, and even end users. It is also a good way to keep a history of problems with machines.

If there is a place where NetSaint is less convenient, it is with the configuration files. The files allow you to have very fine control over the machines being monitored; however, it is not possible to apply a directive to a group of hosts at one time. For example, if you have 180 hosts that you want to monitor for SSHD, you need to enter 180 lines telling NetSaint to do so. The file is also designed for machine parsing, not human parsing, and can be a little hard to read. But you can get around tedious and error-prone configuration file editing with some scripting. An enterprising user came up with a tool that uses NMAP to discover all the services and then output a configuration file of all the hosts and services listed. This is clever and a big time saver, but it isn't a substitute for more expressive configuration file directives. NetSaint would benefit from looking at Mon's approach to configuration.

NetSaint has possibly the best default Web interface, with several very useful views, and a good amount of optional information that can be added. However, it doesn't allow the degree of customization over appearance that many of the other packages provide.

Performance Monitoring Tools

Performance monitoring is an important topic, so we cover it in order to complete the picture of monitoring, and also to contrast it against event monitoring. Due to the short length of this article, we can only touch on the subject briefly, by discussing the motivation for performance monitoring, typical functionality, and one of the most common free packages, MRTG.

Monitoring the utilization and other performance metrics on your systems is important for capacity planning and load balancing. It can help you identify bottlenecks in overall site performance, understand usage patterns, and predict when extra capacity will be required. In contrast to event monitoring, which is typically concerned with a sudden state change, performance monitoring is concerned with describing long-term trends, or providing statistical summaries of system state. The availability statistics that many event-monitoring packages provide is an example of a performance metric and also highlights the relationship between event monitoring and performance monitoring. Many performance-monitoring packages can also be configured to trigger an action if some metric falls below a certain level. For example, a package that is tracking the amount of swap space can generate an alarm if free swap falls below a certain value. This tight relationship between event and performance monitoring is why they are both covered in this article.

MRTG, MultiRouter Traffic Grapher

MRTG is an SNMP-based package, originally designed for monitoring network usage on switches and routers. MRTG is a very popular tool, and many sites may already be using it to watch their network usage. MRTG generates a series of graphs that chart the input and output utilization of ports on switches and routers. This is the most common use of MRTG. However, MRTG has three features that make it especially interesting as a complement to event monitoring: external data sources, rrdtool support, and threshold triggers.

EXTERNAL DATA SOURCES

MRTG can be configured to collect and graph arbitrary pairs of variables served over SNMP. The following URL describes how to monitor server-based metrics over SNMP with MRTG: *http://net-snmp.sourceforge.net/tutorial/mrtg/index.html.*

Figure 5: Demo MRTG Screen

However, sometimes information that you'd like to have is not available fromSNMP. The following URL has links to many such applications: *http://people.ee.ethz.ch/~oetiker/Webtools/mrtg/links.html*.

An example from our site is information from our batch scheduling system about the number of servers currently being used to process batch jobs. Another useful metric is the total number of servers that are available The availability metrics that event-monitoring systems provide usually only measure the availability of individual machines or services, not the availability of the entire cluster. A useful metric would be what percentage of the total site is up at a given time and summary statistics of long-term availability. These can be easily fed into MRTG and graphed using the external data source features.

An issue with external data sources is that they are polled at the same time that MRTG runs its normal SNMP probes, which may or may not be the appropriate polling interval. The next feature we discuss provides a way around this issue.

RRDTOOL SUPPORT

MRTG keeps an internal database of the metrics it has gathered. What rrdtool does is to break out the functionality of this database into a separate tool called the round-robin database tool. This tool keeps a sliding window of the most information gathered – for example, it will only keep the most recent 1000 datapoints, no matter how many you may have actually collected. rrdtool allows you to perform calculations and generate graphs on the data and any calculations you may have made. rrdtool is actually the database back end for the next generation of MRTG; however, recent versions of MRTG can be configured to use rrdtool for its back end.

Because you can insert data into rrdtool independent of MRTG, rrdtool allows much better control over data collection and increases your control over the graphs being generated. MRTG is designed to graph two variables against each other on its graph – this is not always the kind of graph that you are interested in – sometimes you are only interested in a single value being graphed, or you may want multiple values graphed simultaneously. Another issue is that the sampling interval for MRTG may not be the appropriate sampling interval for other services – MRTG performs all the metrics gathering at the same rate. This is especially important if the process of gathering the metrics is expensive or time-consuming: it may be reasonable to gather information about all the ports on a single switch every five minutes, but trying to gather CPU utilization across 180 servers over SNMP every five minutes can be more trouble than it is worth.

If you are willing to invest some time, rrdtool solves both these problems. rrdtool can store arbitrary time series data, and gives many options for creating graphs. As an example, the following image is a graph of processor utilization across our server farm. The source data comes from our batch scheduler (LSF), the data and stored in rrdtool, and a cron job runs rrdtool every 15 minutes to generate an up-to-date graph.

Another advantage of using rrdtool is that it can easily serve as a bridge to the event monitoring package – the event monitoring package can sample the most recently gathered statistics and generate an alarm if something is out of the expected range of values.

THRESHOLD TRIGGERS

MRTG can also be configured to generate alarms if certain metrics go outside an acceptable range. In this capacity, MRTG operates as a fault/event monitor and is a useful adjunct to the main event monitor. Threshold triggers and rrdtool to interface with the event monitor are both effective for tying the event monitoring and performance monitoring systems together.

Figure 6: Sample rrdtool Output

Conclusion

Monitoring is a key aspect of system administration, especially for sites that have 24x7 requirements. Even for sites without such requirements, administrators may be better off having their monitoring package inform them that something is wrong with their systems before a user does. The packages described cover a good range of monitoring requirements, and if used well, should allow an administrator to know what is happening on their site in excruciating detail.

At the minimum, these tools provide an administrator with clear metrics about the performance of their site. Ideally, these tools can clarify broader policies and provide insight into capacity planning and resource utilization.