System and Network Monitoring Requirements - August 2009

By Edgar "Tex" Nielsen

Current systems in place

SPS

- **Zenoss**
- HP UPS **DevMan** server software
- HP ILM Software
- Script to send a daily **sudo** log summary
- Script to send a daily log summary for SPS and SPSN machines
- IceCube Live
- Mark Krasberg custom **DOMHubMonitor, pager,** and **checkdisk** scripts (being integrated into his nagios)
- WxGoose environment monitors
- Mark Krasberg **nagios** for DOMHubs being implemented(also monitors server disk free space)
- certwatch script monitors
- **mrtg**

SPTS

- **Zenoss**
- HP UPS **DevMan** server software
- HP ILM software
- IceCube Live
- Mark Krasberg custom **DOMHubMonitor**, **pager**, and **checkdisk** scripts (being integrated into his nagios)
- Mark Krasberg **nagios** for DOMHubs being implemented (also monitors server disk free space)
- **mrtg**

222 (List may not be comprehensive)

- **Zenoss**
- HP UPS **DevMan** server software
- HP ILM software
- WxGoose environment monitors
- **Ganglia**

- **mrtg**
- Other products or custom scripts may be in use
- Power monitoring at 222 data center
- SATA Beast web interface

Notes

For the most part, all of the above systems are independent of each other and have independent notification systems, most commonly email. The lack of central control inevitably results in more notifications than using one monitoring system with one notification system.

HP UPS **DevMan** server software - emails on UPS issues such as power failures; some SNMP support as well
HP ILM software - HP software (command line and web interface) to monitor Proliant hardware health; some SNMP support as well
WxGoose environment monitors - Environmental monitors of temperature, airflow, humidity; email alerts; SNMP support as well
Ganglia - Similar to **mrtg**, used frequently for statistics of clusters; at 222 used on many hosts
mrtg - RRD-based tool for gathering and displaying statistics; at 222 used for average network statistics

What should be monitored?

I recommend monitoring as much as possible, but only a *subset* of the monitored items generate alerts. Monitoring everything helps staff determine the source of problems when working on outages or the post-outage post-mortem. For example, if alerts start happening on a server and the console shows an administrator on the system, one can contact that administrator and see if they are doing unscheduled maintenance. The IceCube infrastructure is not large enough to cause scalability issues for most monitoring systems. Most items would be checked at five minute intervals, but some will be checked at more infrequent intervals - for example, there is no point in checking the expiration of SSL certificates at any frequency less than once a day. For performance monitoring, five minutes is too infrequent to provide much insight into the actual performance of the system. For performance monitoring, a separate system will be needed in order to support gathering statistics at intervals of less than one per minute. More discussion on performance monitoring is in a separate section below.

Items with ★ are trivial to implement in most network management systems(NMS) systems and so should be in the initial implementation.

Servers

- CPU temperature and fan speeds★

- Load★
- Needed processes★
- Number of processes★
- Memory usage★
- Memory errors★
- Active Users★
- Network Interface stats(IO and error rates)★
- Disk Space and inode usage★
- Disk read-only state★
- Disk IO statistics
- Disk drive SMART status★
- HP RAID array status★
- NFS mounts check★
- System errors such as kernel errors★

IceCube Specific

- DOM status (MK's **DOMHubmonitor**)★
- MK's **checkdisk** script
- Winter-over trip monitor (MK)
- ITS state
- SPADE status(JBOSS status, taping state, etc)
- SPTR status and queue size
- pDAQ services monitoring★
- PnF services monitoring
- CnV services monitoring
- IceCube Live monitoring
- *sps-itfreeze* FCU program monitoring
- GPS checks (possible with our model?)

Network Hardware

- Temperatures★
- Active Users★
- CPU load★
- Configuration changes★
- Network interface stats★

Services

- AMANDA backup status
- DHCP server status★
- tftp server status★

- Print server status★
- RHN up2date status check
- RHN checks (oracle, etc.)
- LDAP status checks★
- DNS server checks★
- NTP server checks★
- MySQL and Postgres health★
- SMTP and IMAP server health★
- Email roundtrip time monitoring★
- PBS monitoring
- NFS server and client★
- automounter★
- Monitor availability of the various web servers★
- JMX monitoring of JVM statistics

Other

- Environmental monitors(WxGoose, USAP equipment) temperature, airflow, door/fire dampers open/closed...★
- Monitoring of Xen Dom0 servers
- UPS state and statistics★
- Tape library state and statistics
- **sudo** working check★
- SSL certificate expiration★
- License expiration - RHN, PBS
- External array monitoring - temperatures, RAID status, IO rates if possible
- HP IP KVM monitoring if possible, though do not think it is possible to detect when it needs to be rebooted
- Monitoring of log files to feed the monitoring system using a tool such as **splunk** or **SEC.** More discussion below.
- Meta-level monitoring of subsystems - such as CnV, PnF, pDAQ. For example, pDAQ would be flagged as critical if any of the supporting processes are not running on the DOM hubs, *sps-evbuilder, sps-expcont, sps-letrigger,* etc.

Policies

- SPS notifications to the North are an issue due to the limited bandwidth available when the TDRSS and GOES satellites are down. Two approaches are:
    1. Treat the winter-overs as Tier-1 support - only they get the initial notifications. If a problem persists over a period of time, automatically escalate and notify the North. Note that the NMS will be forwarding critical alerts into IceCube Live's alert log as well so staff in the North can monitor that log if interested. In addition, a periodic summary of alerts can be sent to the North. Ideally, I would want the monitoring history to be

synchronized to the North as a read-only copy when TDRSS is up. The feasibility of this is highly dependent on the NMS.
2. Always include the North in notifications as is the current practice. Due to the limited bandwidth, I would want only one email addresses in the North allowed which would be an email mailing list where interested users could subscribe to the list. In addition, the mailing list software must support archiving and provide an RSS feed so interested users could subscribe to the RSS feed with a standard RSS reader. For example, during the recent switch failure at the ICL(which latest about four hours), **Zenoss** alone sent over 2000 email messages to the North which would have caused a major backlog on the low-bandwidth Iridium email system. This backlog would slow messages from the North to the winter-overs as well as causing email delays for other NPX users.

- Mail from monitoring systems should never be filtered by recipients as this practice almost always ends up with the user missing outage notifications:
    - if being filtered due to too many messages - NMS needs tuning
    - if messages not relevant to the user - either NMS needs tuning or recipient should not be getting the emails at all
    - if only want to see them after an outage - have emails go to a mail archive system for people to review or use the NMS console to review

Feature Requirements

For the most part, most NMS packages provide the same set of functions. **Zenoss** and **nagios** support similar functions - as a commercial product **Zenoss** is more smoothly integrated and somewhat easier to install while **nagios** has much more documentation and wider support community. For example, **Zenoss** provides graphing of monitored items out of the box while with **nagios,** one must choose between several different graphing packages and then do the integration. Of course, one can get prepackaged **nagios** distributions that have already done this integration.

Agent and Agentless monitoring

The two primary approaches to the monitoring of computers and other hardware are *agent* and *agentless* monitoring. Agent-based systems require the installation of client software on each computer while agentless systems generally require the monitored systems to have an SNMP agent running. Some systems - such as **nagios** and **Zenoss** - can do both. In my experience, hosts are best monitored with agents while other devices are best monitored using SNMP. For example, many linux distributions come with very outdated SNMP agents.

Dependencies

To reduce the number of needless notifications, the NMS must be aware of the network topology as well as support dependencies on both hosts and services. For example, each

rack in the ICL has a dedicated network switch which connects all devices in the rack to the core switch. If this rack's switch is turned off, the NMS must generate only an alert on the outage of the network switch, and no alerts about the *downstream* devices being unavailable. In the NMS console, the state of these downstream devices will be displayed as unknown.
A service dependency example would be the automounting of home directories. The automount table is kept in LDAP, so if both LDAP servers are down, the automounter will fail. As the real problem is with the LDAP servers, no reason to send notifications about the automounter not working on each system.

Configurable views

An NMS should different views into the data. For example, one might wish to see events of only a certain level or only the state of systems in a given rack. For example, Kael Hanson may wish to only see DOM hubs and ***pdaq*** machines. Using different views helps pinpoint issues more quickly. Views may or may not be configurable per user. Some systems also support custom image maps for presentation such as having pictures of each rack.

Notifications

The NMS must support a variety of notification methods. Email is normally the default, but one must be able to also use custom scripts for maximum flexibility. While one can often *hack* out a solution with email notifications using a tool such as **procmail,** such systems would introduce additional points of failure in the notification process. For example, I would recommend sending pages to on-call staff via pagers or SMS using a modem rather email which relies on many other systems. Notification systems need to support escalation. A simple example is if the primary winter-over does not acknowledge an alert within 30 minutes, the backup winter-over will receive notifications. If the backup winter-over does not acknowledge, then one could have someone in the North contacted.

Meta-level monitors

Ideally, an NMS lets one define high level monitors that are actually just reporting on the state of several real monitors. High level monitors are often called *business process* monitors. For example, a **pDAQ** monitor would only be in a good state if all of the pdaq processes on *sps-expcont, sps-evbuilder, sps-letrigger,* and all the DOM hubs were running. A SPADE monitor might only show a good state if no errors have been logged in 30 minutes, files are being transferred to/from the *sps-sattx* machines, and no tapes need to be inserted.

Templates

In order to simplify the NMS configuration, *templates* must be supported. One defines templates for groups of systems which are similar and perform identical functions. For example, one could define a template for DOM hub, and configure the monitoring characteristics for a DOM hub. Then one creates objects for each DOM hub which depend on the master DOM hub template. If one later decides that DOM hubs should have a new item monitored, one simply modifies the template to enable the new item monitoring on all 59 DOM hubs. Without templates, the configuration of the NMS become much more difficult and error prone as one has to manually edit many objects instead of just the template.

Change Management/Revision Control

One must be able to track changes to the NMS to ensure no unauthorized changes occur as well as accountability for authorized changes. Systems with text-based configuration files can also be placed under source code control. In an environment such as IceCube, multiple people will be working with the system and may accidentally introduce conflicting changes in the configuration. Similar issues exist for system administration, network administration and software development!

Easy to write plugins/addons

Most monitoring systems have methods for users to write their own plugins/addons. Custom plugins are essential for monitoring services/hosts unique to the organization. Monitoring the DOM hubs is one example unique to IceCube. Ideally the API supports multiple languages including perl, python, and Bourne shell.

Support/User Community

The chosen NMS should have commercial support options available which should be evaluated prior to purchase. In addition, the product should have a large and active user community with forums and third party plugins/tools. Finally, the availability of multiple books and training courses helps indicate the long term stability of the product.

Nice To Have

- Text-based interface, or usable via text-based web browser - when working in emergencies one may not have a graphical interface available. In addition, while a web interface works reasonably well across the satellite using an NX remote desktop, it is often more convenient to use text based tools.

Software Products

Zenoss

Zenoss defaults to monitoring only by SNMP but does support using agents via ssh on the monitored systems. IceCube uses the enterprise version of **Zenoss,** but a free version is available as well. I have never found a comprehensive list of the additional features of the enterprise version but these include:

- Enterprise ZenPacks
- Enterprise Reports
- User ACLs for dashboard access

The package provides integrated graphing of monitored items, and somewhat customizable dashboards. **Zenoss** is currently implemented at SPS, SPTS, and at 222. The 222 implementation is the most mature of the three instances, with SPTS having the least mature implementation. Both SPS and SPTS implementations need quite a bit of work to bring them up to the state of the 222 instance. The 222 instance is a work in progress and needs additional configuration to monitor more services and hardware. For example, I do not believe that any of the HP Proliant management software components are monitored except for temperature; status of the onboard RAID is one such component. The HP software *may* be sending traps to **Zenoss** on errors. In addition, the 222 instance needs tuning to reduce the rate of false events. For example, last Saturday and Sunday(08/08-08/09), the 222 instance of **Zenoss** shows quite a few events:

Level	Number of Events
Critical	31
Error	769
Warning	689
Total	**1489**

Zenoss generated 4155 email messages during this period. Excluding the 3725 messages for Paul Wisniewski still leaves a total of 430 messages split between Steve Barnet, David Bogen and Paul McGuire. The *Computing News* web page only shows one problem for the weekend - the PBS license expiration. This leads me to believe substantial tuning is required for the 222 instance.

Zenoss supports our requirements, though the details of how service dependencies, configuration management, and meta-level monitoring are implemented would need more in depth investigation. I believe David Bogen, Steve Barnet and Karthik Pandian have experience with **Zenoss.**

Nagios

Nagios supports both agent and agentless modes of operation. Normally configuration is via text files but a variety of configuration GUI packages are available, though I believe text files are superior for managing configuration. **Nagios** has a very large user community, several books, commercial support is available and large numbers of plugins are available. The only current IceCube **nagios** implementation is Mark Krasberg's **nagios** SPTS. The implementation is similar to my implementation at SPS last year, with the addition of **nagios** versions of Mark Krasberg's **DOMHubMonitor** and **checkdisk** scripts. His implementation is currently using **ssh** based checks, but will be switching to a more scalable system in the near future.

Nagios supports all of the requirements listed. The new winter-overs as well as Mark Krasberg, Tex, and Victor Bittorf have varying degrees of experience with **nagios.** I believe Karthik Pandian also has some **nagios** experience.

Other Potential NMS Packages

- **Zabbix -** open-source package with integrated graphing
- **OpenNMS -** open-source package primarily for SNMP only monitoring, though collection via JMX and HTTP has been added. Integrated graphing
- **Reconnoiter -** new open-source package still under development. An attempt to make a highly scalable best of breed monitoring system.Potentially useful for both system monitoring and performance monitoring. This package looks quite interesting, but is considered to be an alpha release.
- **SNMPc -** quite nice commercial package for SNMP only monitoring. I have mainly used this as an adjunct to my primary NMS.
- **Hyperic -** commercial package which - like **Zenoss -** has a free version. Also like **Zenoss,** quite a range of opinions on this package from great to horrible. Recently bought by a company which was just bought by VMWare. This was the primary package I recommended investigating to Darryn in 2007 (along with **Zenoss)** if **nagios** was not implemented. The free version has the significant shortcoming of not supporting templates so configuration management is probably too much work.

Recommendations(still being written)

While some of the other NMS packages are interesting, I recommend that IceCube choose between **nagios** and **Zenoss** as evaluating yet another package would take quite some time. Both packages can fulfill the monitoring needs of IceCube - the main differences are in pricing and level of *out of the box* integration. If **Zenoss** is preferred, it might be useful to test the free version to determine if the function is adequate as I am not sure IceCube is using many of the features of the Enterprise versions.

I do have a preference for **nagios** as I have a fair amount of experience implementing this package but **Zenoss** appears equally capable. In a rough sense, one could say that **Zenoss** is to **nagios** as MacOS X is to Windows. Both get the job done, with **Zenoss** having a more smoothly integrated browser based GUI while the **nagios** browser GUI is not nearly as smooth. Then again, **nagios** also has more than one possible browser based GUI as well as text based ones.

In the end, all monitoring systems still require training and frequent tweaking to keep the system in sync with reality. With some work, configuration management systems can help with some of these tasks - for example, a newly created system could be automatically add into the monitoring system. At SPS, I would recommend that the winter-overs being given at least some of the day to day responsibility for the NMS, while at Madison the day to day responsibility could be shared - though *at least* one person should be given the time/training/resources to become an expert with the system.

Complementary Tools

- **RANCID** - network switch configuration monitor which both backs up and tracks changes to configuration and hardware
- **smokeping** - visualization and monitoring of network latency
- **atop** - substantially enhanced version of **top** that can run as a daemon and is useful for a *looking back in time* version of **top**. In addition to CPU and processes, tracks disk and network IO in one display.

Log Monitoring and Analysis Tools

Log analysis will be an important part of any monitoring system as many subsystems are primarily visible only via log files. Ideally these tools are run on the central log server rather than instances on each system. A key issue with log monitoring is that no list of all possible log messages exists. So one must setup the monitor to alert on known critical messages, filter out known non-critical messages, and warn on previously unknown log messages. As unknown messages are found, they must be categorized into critical or benign. In addition, the log monitor must be flexible - i.e. a log message that appears once might not be a problem, but if the message appears X times in Y minutes - that might be something to generate an alert into the monitoring system. In addition, non-UNIX operating systems such as Microsoft Windows need to configured to send log messages to the IceCube **syslog** servers; often this is only possible with third party tools.
I would not be surprised to end up using several tools, one for monitoring and one for analyzing logfiles. Many tools exist for log monitoring and analysis, some possible candidates are:

Security Event Correlator(SEC) - Perl-based log monitor that is very powerful and configured via text files. Works very well and highly flexible.
splunk - commercial GUI tool which is very popular in the commercial sector. I have not personally used **splunk**, but it has a very good reputation. Free version limited to indexing 500MB of logs per day and cannot generate alerts. Enterprise version cost is based on the amount of data indexed each day.
sisyphus - log analysis tools by Sandia National Labs which attempts to automatically detect *interesting* log entries. Martin Merck learned about this product, I have not investigated this product.

Performance Monitoring

Performance monitoring needs to collect statistics at intervals from 1 to 60 seconds in order to provide the detailed information to analyze problems. The graphing done by the NMS can help guide which systems need to be monitored at a higher resolution. IceCube has experience using **mrtg** and **ganglia** in this role. Depending on the tools used, it may be easier to gather this information at 1 minute intervals and use sub-minute intervals only when looking at specific issues. Alternatively, one could rely on the NMS five minute interval collection and only use tools at a finer resolution when actively debugging issues. This route is probably better as data collection at high resolution generates a considerable amount of data.

> **collectd -** similar to **ganglia,** designed for low overhead and defaults to a ten second interval. Unlike **ganglia, collectd** only generates RRD files. One uses any RRD graphing tool such as **drraw** for graphs. I have not used this product.
>
> **collectl -** Perl script from HP which is designed for low overhead and also defaults to ten second intervals. **collectl** also does not graph but can be fed into **ganglia.** HP also has a graphing component available to customers. I have used this software a fair amount and have found it quite useful for performance monitoring at one second intervals
>
> **ganglia -** very well known tool, especially in HPC circles. I am not certain if **ganglia** supports intervals less than 1 minute in length.