Monitoring systems: Concepts and tools

Zdenko Škiljan Branimir Radić

Department of Computer Systems,
University Computing Centre, Croatia
{zskiljan, bradic}@srce.hr

Abstract

Every computer system has to be systematically supervised on account of recognition of critical circumstances that need troubleshooting, system/application tuning or, in the end, system upgrade. During past years, various tools were developed for that purpose on UNIX/Linux platform. This presentation brings an overview of both traditional tools for monitoring UNIX/Linux systems as well as of complex tools for monitoring distributed systems.

1 Opening remarks

Among traditional tools, different sets of tools will be considered according to their specific application area; basic system monitoring tools, system integrity monitoring tools, system performance monitoring tools and services activity monitoring tools. Data visualization through the monitoring system is crucial in long-term diagnostics and decision-making, so the most prominent solutions in this area will be commented. Finally, special attention will be paid to the concept of cluster monitoring and its fundamental principles.
This is not a comprehensive review of tools. Due to limited space, it merely takes into consideration selected prominent Open Source solutions in different fields of monitoring.

We shall consider different aspects of traditional approach to monitoring, oriented towards a single system, as opposed to a cluster oriented monitoring approach.

2 Single System Monitoring Facilities

We shall consider the following single system monitoring facilities:

- Log files
- /proc pseudo file-system
- Basic OS commands for system monitoring
- File integrity checkers
- IDS systems
- System performance monitors
- Service activity monitors
- Visualization tools

As the system comes into use, the very first help to a system administrator are log files and basic system commands, convenient to make the system tuning and debugging easier.

These commands are building blocks of system activity monitors that administrators use to keep an eye on a system.

2.1 LOG files as a source of information

Log messages are the most valuable sources of information for system administrators.

Most UNIX-like systems provide an API for application programs, which sends log messages to the system, where they can be centrally handled at the discretion of a system operator.

Applications and their messages vary significantly in their importance to certain audience. If a number of applications are

considered "critical" and their status is the system administrator's responsibility, he doesn't want to search to find out where and how every critical application logs its status.

That is where Syslogd comes in as a precious information collector and distributor. Syslogd is a daemon, which listens to various system and application messages and distributes them according to the user-defined settings in syslogd.conf file, so they can be easily analyzed with respect to their priority.

2.1.1 /proc pseudo file system

/proc pseudo file system is an interface to kernel data structures, which is mounted as a regular file system.

The /proc mount point provides a reasonably complete set of local monitoring data, including system load, memory utilization and per-process system resource utilization, some general information about a system and as such is a very useful source of information about the state of a system.

2.1.2 Basic OS Commands for System Monitoring

There are numerous commands that help to supervise the system activity.

Process examination: ps, top
Network examination: netstat, ping, traceroute, ntop, tcpdump, ip.
File system examination: df, free, find, locate, lsof, grep.
System devices examination: iostat, vmstat, sar.
General information: uname, uptime, linuxinfo

Every system administrator should have knowledge of the way these commands are used.

2.2 System log anomaly monitors

To be able to excerpt the most useful information from all log files, many utilities were developed that help with log analysis.

2.2.1 Logcheck (LogSentry)

Logcheck [1] is a general-purpose tool that is designed to run on regular basis and check system log files for security violations and unusual activities.

It has a huge database of predefined patterns and it simply sends a mail message with logs that match these patterns. Administrators can easily modify patterns database.

Advantages: Easy to use, powerful predefined database of patterns.
Disadvantages: Best performance requires good knowledge of regular expressions, limited alerting mechanism (mail only), it is not a real-time monitor.

2.2.2 Simple WATCHER

Simple WATCHER (Swatch [2]) is as Logcheck a program used to parse through multitude of log data generated by various programs.

It is fully configurable with triggers (actions), so that while it is continuously monitoring log data in "real-time", it can take actions based upon certain high-priority events administrator decided to watch for.

Advantages: Easy to use, refined action configuration facilities, performs real-time checking
Disadvantages: Poor database of predefined patterns

2.3 File integrity checking

Integrity checking tools are used for periodical verification of selected files.
They work by first creating a database of unique identifiers for each file or program to be verified.

Integrity checking includes verification of checksum numbers generated by encryption algorithms from the file's contents, but also can include verification of attributes, such as permissions and file sizes.

2.3.1 Aide and Tripwire

Aide [4] and Tripwire [3] are similar Open Source programs designed to monitor changes in a key subset of files identified by administrator, and report on any changes in any of those files.

Files are scanned periodically (daily or more frequently), that is defined through cron facility.

Any change, addition or deletion is reported by mail, so that a proper action can be taken.

2.3.2 AIDE Repository Management Suite (ARMS)

ARMS [5] is a centralized version of Aide, developed by CARNet. It uses client-server model to send aide reports to a central repository where they can be analyzed in a broader context and isolated from original host in case of a break-in.

2.4 IDS systems

2.4.1 Snort

Snort [6] is an advanced system for intrusion detection that utilizes preprocessors and database of predefined patterns to analyze real-time network traffic.

Advantages: Good resource of attach/intrusion information.
Disadvantages: Too many information by default, no inherent response mechanism, imposes too much load on a system.

2.4.2 Snort central

Snort central [7] is a centralized version of Snort, developed by CARNet. It uses client-server model to send Snort logs to a central log repository where they can be analyzed in a broader context. It uses a web interface to formulate queries on collected data and gives an overall security status of whole monitored network.

2.4.3 PortSentry

PortSentry [8] is a IDS system which actively protect against portscans. It detects portscans and can be configured to log them and even block offending hosts, making completion of a port scan difficult.

2.5 System performance monitors

Function of a performance monitor is to collect data from different layers of the system — hardware, kernel, service, and application layers — for administrators and users who collect such data for diagnostic purposes in order to fine-tune the performance of different system layers.

2.5.1 SNMP

One of the most widely used approaches to network monitoring and management is Simple Network Management Protocol (SNMP). SNMP has two distinct parts, a format for describing data about computers and a protocol for transmitting that data over a network.

Main problem with SNMP is that it has suffered some serious security flaws.

Advantages: Powerful, feature rich.

Disadvantages: It suffered serious security flaws, not simple to use.

2.5.2 Orca/Orcallator/Procallator/Orca Services

Orcallator and Orca Services [9] collect system data and prepare it for Orca. Procallator is a Linux version of Orcallator.

While Solaris version explores SE toolkit kernel interface, Linux version relies on information fetched from the /proc pseudo file system.

Orca uses collected data to make very useful graphs through RRDTool. With rsync or ftp data can be collected from multiple hosts and visualized.

Advantages: Offers very detailed system data, convenient for long-term decision- making as well as detection of current problems and bottlenecks.
Disadvantages: Not Scalable, Uses RRDTool which requires compatible systems.

2.6 Service activity monitors

2.6.1 Orca services

Orca services [9] are basically Orcallator rewritten to be able to monitor system services like smtp, pop3, http, dns, radius, etc. It reuses Orcallators code to provide appropriate graphs.

2.6.2 MON monitoring daemon

Mon [10] is a general-purpose scheduler and alert management tool used for monitoring service availability and triggering alerts upon failure detection. *Mon* was designed to be open and extensible in the sense that it supports arbitrary monitoring facilities and alert methods via a common interface, all of which are easily implemented with programs in C, Perl, shell, etc., SNMP traps, and special *mon* traps.

Advantages: extensibility (monitors, alerts, user interface), portability (Perl), simple usage.
Disadvantages: Not refined data collection on client side.

2.6.3 BigBrother

BigBrother [11] is a monitoring tool very similar to MON monitoring daemon with some disadvantages – like limited subset of network protocols supported and limited notification methods.

2.7 Visualization Tools

2.7.1 Multi Router Traffic Grapher (MRTG)

MRTG [12] is a powerful C/Perl based data collector and data grapher.

It is a system designed to store and display time-series data (i.e. network bandwidth, machine-room temperature, server load average).

It basically does three things:

- Fetches counters from devices,
- Calculates the rate and store this for future use,
- Draws a picture from the stored information

Its main limitation is that only two counter types can be monitored in each graph.

Advantages: Simple and powerful graphing tool
Disadvantages: Limited amount of counter types in one graph (two counters only).

2.7.2 RRDTool

RRD is the acronym for Round Robin Database.
RRDTool [13] is a reimplementation of MRTG graphing and logging features. It is much faster and more flexible then MRTG.
It stores data in a very compact way that will not expand over time (it's not scalable).
It presents useful graphs by processing data to enforce a certain data density.
While MRTG generates the traffic graphs in a periodic basis (normally each 5 minutes), RRDTool generates them only when requested (on demand).
Advantages: Powerful, feature-rich
Disadvantages: Complicated, database not scalable, database not platform independent

2.7.3 JROBIN

JRobin [14] is a Java version of RRDTool that offers benefits of Java portability. Using the same logic, concepts and definitions, it provides the very same output for the same RRD input.

Advantages: Portable, database is platform independent and scalable.
Disadvantages: Complicated as RRDTool.

2.7.4 Perl GD::Graph

GD::Graph is a perl5 module used for creation of various chart types, such as bar charts, pie charts, line charts etc.

It relies on GD library, which is an ANSI C library for dynamic creation of images. Among other formats, GD creates PNG, JPEG and GIF images.

Advantages: Uses any type of database, database can be created dynamically.
Disadvantages: Difficult to use (it requires programming effort), limited documentation, reduced graphing features.

3 Cluster Monitoring Tools

Traditional monitoring systems are usually single system oriented, designed to work in a limited environment. They typically do not include built-in mechanisms for active response except a primitive mechanism for sending alerts to administrators.

Cluster concept, on the other hand, demands employment of monitored data for efficient job distribution and centralized access to status data in order to make access easier to the administrator.

As monitoring function is inherent to JMS, a review will be made of cluster monitoring devices comprised in:

1. JMS built-in monitoring
2. Self-sufficient Cluster Monitoring systems

3.1.1 JMS built-in Monitoring

Based on current nodes state and in order to achieve maximal utilization of cluster resources, JMS distributes jobs within the cluster.

JMS consists typically of three components: Queuing Module, Scheduling Module and Resource Manager.

Queuing module accepts job requests and distributes jobs into queues. It then sends request to Scheduling module to process queued jobs. Its function is also to maintain job allocation and resource consumption data.

Scheduling module makes a decision on which nodes and by which priority jobs will be distributed. Decisions are based on the job resource request (obtained form Queuing Module), job management policy defined by administrator (bigger, smaller, parallel etc. jobs) and resources state (given from Resource Manager Module).

Resource Manager Module collects and maintains node status information and performs actual job execution and supervision. It consists of client and server components. Server component collects client's node information. Client component does actual nodes data collection and passes data to a server.

Dedicated cluster systems usually do not employ any special **job distribution based** on monitoring, but rather **on a predefined policy and minimum node status information**.

Job distribution based on monitoring (load balancing) is typically used in case of non-dedicated clusters.

3.1.2 Independent Cluster Monitoring Systems

Cluster monitoring systems like Ganglia and Supermon are fully independent of cluster type or JMS type and consist of two major entities: server that collects cluster state information and a GUI-based front-end, which provides system activity visualization.

3.1.2.1 Ganglia

Ganglia [15] is a java based, scalable distributed monitoring system for high-performance computing systems such as clusters and Grids.
Ganglia includes following components:

- Gmond – local monitoring system
- Gmeta – wide-area monitoring system.
- Ganglia web front-end

Gmond operates on cluster level and uses UDP multicast to exchange data within a cluster.

Each cluster node listens and accepts data from neighboring nodes providing functionality for easy adoption of new nodes in the cluster and data storage redundancy as every node contains state information from other nodes.

Gmond communicates with Gmeta using XML stream over TCP.
Gmeta processes and presents monitoring information gathered from one or more clusters.

Gmond has four main tasks:

- Monitors changes in host state,
- Multicasts relevant changes,
- Listens to the state of all other ganglia nodes via a multicast channel and
- Answers to the requests for an XML description of the cluster state.

Ganglia web front-end presents all historical data saved in Round-Robin databases by Gmeta in HTML, allowing all cluster, hosts and host metrics to be viewed in real-time.

Advantages: New nodes can be added easily, it is portable, it has a fine web interface, it is widely used and can be easily integrated with other monitoring systems (e.g. Globus MDS, MonALISA).

Disadvantages: Database is not scalable (as it's using RRDTool)

3.1.2.2 Supermon

The Supermon [16] is another distributed monitoring system, slightly different from Ganglia.

Supermon consists of three different components:

- A loadable kernel module providing data (through /proc entries in form of s-expressions)
- A single node data server (Mon) that serves data prepared by the kernel module,
- A data concentrator (Supermon), which composes samples from many nodes into a single data sample through a TCP port.

Single node server and data concentrator allow clients to retrieve data samples through TCP socket.

Opposite to Ganglia, Supermon gets a single node collected data on demand. It also must have knowledge of all cluster nodes.

Supermon uses a modified version of the SunRPC rstat protocol to collect data from remote cluster nodes.

This modified protocol is based on symbolic expressions (S-expressions) instead of XML with advantage that they operate in a heterogeneous environment, using plain text and variable size expressions. Their main disadvantage is that the used language is not a standard language like XML.

Advantages: Fast and efficient data-collector.
Disadvantages: New nodes cannot be included in monitoring automatically, poor documentation.

3.1.2.3 Hawkeye

Hawkeye [17] provides a simple and lightweight way for system administrators to monitor and manage distributed systems. Hawkeye is designed by Condor [18] group and mainly used for monitoring Condor pools.

Hawkeyes architecture comprises of four major components:

- Hawkeye pool,
- Hawkeye manager,
- Hawkeye monitoring agent, and
- Hawkeye module.

Pool is a set of computers, in which one computer serves as the Hawkeye Manager and the remaining computers serve as Hawkeye monitoring agents.
Hawkeye Manager is the head computer in the pool that collects all monitoring information and also handles all user queries.
Hawkeye Monitoring Agents send ClassAds to the Manager at specified intervals.

Classified Advertisements (*classads*) are the *lingua franca* of Condor. They are used for describing jobs, workstations, and other resources. They are exchanged by Condor processes to schedule jobs. They are logged to files for statistical and debugging purposes and are used to enquire about current state of the system.

Advantages: Multiplatform system, possible custom-made sensors.
Disadvantages: Poor front-end, under development.

3.1.2.4 Other Systems

There are many other monitoring systems currently available and in use. We will briefly mention some of them:

NetLogger [19] is designed and used to monitor behavior of different elements of the application-to-application communication path in real-time, in order to determine location of time-consuming bottlenecks within a complex system.

Using NetLogger, distributed application components are adapted to produce time-stamped logs of "interesting" events at different critical points of a distributed system.
Events from each component are correlated, which allows one to characterize in detail the performance of all aspects of the system and network.

The NetLogger Toolkit itself consists of four components:

- An API and library of functions to simplify the generation of application-level event logs,
- A set of tools for collecting and sorting log files,
- A set of host and network monitoring tools, and
- A tool for visualization and analysis of the log files.

In order to instrument an application to produce event logs, application developer inserts calls to the NetLogger API at all critical points in the code and links application with NetLogger library.

NetLogger ability to correlate detailed application instrumentation data with host and network monitoring data has proven to be a very useful tuning and debugging technique for distributed application developers.

Paradyn [20] does performance monitoring for long-running parallel and distributed applications and adapts the performance of these applications by instrumenting them dynamically, at run-time, using the monitoring information it collects.

Falcon [21] is an application specific on-line monitoring system that provides its own set of instrumentation libraries and controls, which developers of applications can use to tune its performance.

/dproc [22] – a distributed pseudo file system, which extends /proc interface with resource information collected from both local and remote hosts to predictably capture and distribute monitoring information; dproc uses a kernel-level group communication facility, termed KECho, which is based on events and event channels.

4 Conclusion

There is a multitude of different monitoring tools, which are used for different purposes. Many of the above-mentioned monitoring

systems are inevitable on every computer system; especially security related ones, while others are useful in different environments, which depend on user requirements and limitations as well as system-specific features.

When planning cluster monitoring system, and in order to be able to utilize cluster inherent facilities as well as self-contained monitoring systems to obtain maximal amount of crucial information with minimum impact on system performance, one should carefully think about cluster type to be used and its specific features.

5 References

[1] Logcheck: http://sourceforge.net/projects/logcheck/
[2] Swatch: http://sourceforge.net/projects/swatch/
[3] Tripwire: http://sourceforge.net/projects/tripwire/
[4] Aide: http://sourceforge.net/projects/aide
[5] ARMS: ftp://ftp.carnet.hr/pub/debian/dists/carnet/opt/binary-i386/
[6] Snort: http://www.snort.org/
[7] Snort Central: ftp://ftp.carnet.hr/pub/debian/dists/carnet/opt/binary-i386/
[8] PortSentry: http://sourceforge.net/projects/sentrytools
[9] Orca: http://www.orcaware.com/orca/
[10] MON: http://www.kernel.org/software/mon
[11] BigBrother: http://bb4.com/
[12] MRTG: http://people.ee.ethz.ch/~oetiker/webtools/mrtg/
[13] RRDTool: http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/
[14] Jrobin: http://www.jrobin.org/
[15] Ganglia: http://ganglia.sourceforge.net/
[16] Supermon: http://supermon.sourceforge.net/
[17] Hawkeye: http://www.cs.wisc.edu/condor/hawkeye/
[18] Condor: http://www.cs.wisc.edu/condor/
[19] NetLogger: http://www-didc.lbl.gov/NetLogger/
[20] Paradyn: http://www.cs.wisc.edu/~paradyn/paradyn.home.html
[21] Falcon: http://www.cc.gatech.edu/systems/projects/FALCON/
[22] Dproc: http://www.cc.gatech.edu/~sandip/research.html