Requirements for the Monitoring System of the italian BaBar Reprocessing Farm

Moreno Marzolla, *moreno.marzolla@pd.infn.it*

Università "Ca' Foscari" di Venezia and INFN Padova, Italy

4th Draft, 16 October 2001

This document contains a high-level overview of the requirements for the monitoring system of the italian Reprocessing Farm which is being built in Padova for the BaBar experiment. No implementation detail is given at this time.

Contents

1 Revision History

oct 3 2001, First Draft

Initial version.

oct 4 2001, Second Draft

Included comments from Alvise Dorigo.

oct 12 2001, Third Draft

Included a list of software tools in the references; also, changed slightly the names of the monitoring layers.

oct 16, 2001, Fourth Draft

Added Glossary.

apr 17, 2010

Fixed some typos.

2 Introduction

The Reprocessing Farm for the BaBar experiment, which will be installed at INFN Padova, will consist of about 150 dual-processor machines. Each year this number will approximately double. The farm will be connected to 15 TB of disk space, and to a tape library.

The room hosting the farm will have air conditioning devices providing a suitable temperature and humidity for the correct operation of the farm. Sensors for reading the temperature and humidity of the room will be available. A UPS (Uninterruptible Power Supply) device will protect the system from short unexpected loss of electric power.

Such a system needs a suitable monitoring infrastructure in order to check its correct behavior and to inform the administrators of any problem. The monitoring system should be flexible enough to meet future changes of the requirements, should be scalable and should tolerate a *moderate* amount of failures, both of the monitored system and the monitoring infrastructure itself, without stopping to work.

3 Requirements

3.1 What needs to be monitored

The monitoring system should check the *environmental condition* of the room hosting the farm, the *hardware status* of the machines (including the network switch and the tape library), and the *software status* of each machine.

The Environmental Monitor should check:

- The temperature in the computer room;
- The humidity in the room;

The Hardware Monitor should check the internal hardware sensors of each computer, and report:

- Fan speeds;
- CPU temperatures;
- Whether a machine is down;
- Whether some of the local disks failed;
- Whether some of the local network cards failed;
- Whether one of the CPUs failed;
- Whether the air conditioning system is up and running;
- Whether the UPS is up and running.

The Software Status Monitor should check, for each machine:

- The utilization of each CPU, possibly showing the amount of CPU time spent in user and system mode;
- The I/O rate for the local disks (how many bytes per second are read/written at the monitoring time instants) for the relevant machines; it is assumed that clients will make little or no use of their local disks, so this information is especially important for servers.
- The system load (number of processes in the Ready queue over a fixed time interval);
- The total physical memory usage;
- The available disk space on each machine, especially on the AMS server and Oprserver; in particular, special attention should be paid to disk partitions holding directories containing temporary files and/or log informations (`/tmp`, `/usr/tmp`, `/var/log` and so on), as these partitions are more likely to fill up.
- The utilization of the network as reported by the network switch; also, it could be interesting to check which fraction of the network load is due to UDP or TCP connections.
- The utilization of the network for each machine (how many bytes per second are transferred to/from the network card);
- The percentage of packet losses;
- Whether the server processes are running or crashed on their machines. Examples of server processes are the AMS server, Lock Server, OID Server, PUD Daemon, Clustering Hint Server.
- How many Elf are running;
- The statistics related the tape library usage (number of bytes transferred per second to/from the library, number of tape mounts/unmounts, number of free tapes etc.);
- The number of standing and pending locks reported by each Lock Server process;
- The number of active transactions on each AMS process.

3.2 General Monitor structure

The monitoring system should be based on open and widely-used protocols, standards and languages, whenever they are appropriate. The system should be well documented, and the documentation should be kept up-to-date with the actual code.

The monitoring system should not consume too many resources (CPU time, network bandwidth) to perform its tasks. If possible, a formal performance evaluation of the monitor should be performed, in order to know in detail which fraction of the resources are used by the monitor, and its impact on the running system should be understood in as much detail as possible.

The monitor should be scalable, that is, it should provide reasonable response times no matter how the system will be expanded in the future. It should also be reasonably fault tolerant: the monitoring system should continue to work even in presence of hardware and/or software failures, both on the monitored system and on the monitoring infrastructure itself. The degree of tolerance to failures should be chosen to obtain a reasonable balance between system complexity and fault resiliency.

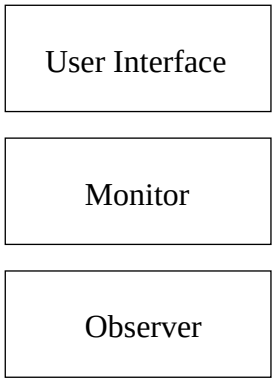

Figure 1: Layered structure of a generic monitoring system.

The monitoring system should support different user interfaces; ideally, it should be built according to a "layered" architecture to provide a general interface over which any kind of UIs can be built. It is essential that a set of command-line tools be available; a graphical interface is certainly useful, but initially not mandatory. The following figure illustrates a simplified view of the monitoring architecture:

In the figure above, the lower layer (*Observer*) deals with the physical acquisition of the status informations from the various devices; this layer should know how to interact with the observed device in order to obtain the monitoring informations. Such informations are the passed to the *Monitor* layer, which gathers monitoring informations from various sources and distributes them to the user interfaces, illustrated in the figure as the *User Interface* level.

The monitor, once started, should require very little or no further intervention from the administrator, apart from periodic checks. The monitor should be configurable to alert the administrator when important events occur. The kind and severity of such events should be user-configurable. Examples of critical events include, but are not limited to:

- Failure of a whole machine
- Disk failures
- Process crashes
- No space left on disk (e.g., `/tmp` full)

The monitoring system can be used also to monitor the utilization of the resources. Hence, the administrator could be notified when the CPU utilization on some host becomes too low (or too high), when the network is near to saturate, or when the available disk space is less than a given threshold.

Data collected by the monitor could be stored for further analysis, especially for performance evaluation purposes. Such informations could be stored away either in raw form, or in a more structured form such that they can be stored in a database.

4 References

4.1 Papers

[stal01]

William Stallings, *SNMP, SNMPv2, SNMPv3 and RMON 1 and 2*, Third Edition, Addison Wesley, 2001

[jain91]

Raj Jain, *The art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*, John Wiley & Sons, 1991

[gridperf]

Grid Performance Working Group Home Page http://www-didc.lbl.gov/GGF-PerfWG/

[subr00]

R. Subramanyan, J. Miguel-Alonso and J. A. B. Fortes, *A scalable SNMP-based distributed monitoring system for heterogeneous network computing*, Proc of IEEE High Performance Networking and Computing Conference, 2000, available online at *http://www.sc2000.org/Proceedings/techpapr/papers/pap280.pdf*

[bear96]

M. J. Bearden and R. P. Bianchini, *Efficient and Fault-Tolerant Distributed Host Monitoring Using System-Level Diagnosis*, Proc. of the 1996 IFIP/IEEE International Conference on Distributed Platforms, available online at *http://citeseer.nj.nec.com/bearden96efficient.html*

[busk93]

R. W. Buskens and R. P. Bianchini, *Distributed On-Line Diagnosis in the Presence of Arbitrary Faults*, Proc. of the 23rd International Symposium on Fault-Tolerant Computing, available online at *http://citeseer.nj.nec.com/buskens93distributed.html*

[bian92]

R. Bianchini and R. Buskens, *Implementation of On-Line Distributed System-Level Diagnosis Theory*, IEEE Transactions on Computers, Special Issue on Fault Tolerant Computing, May 1992, available online at *http://citeseer.nj.nec.com/bianchini92implementation.html*

[mans92]

M. Mansouri-Samani, M. Sloman, *Monitoring Distributed Systems (A Survey)* Imperial College Research Report DOC92/23, 22 September 1992, available online at *http://citeseer.nj.nec.com/mansouri-samani92monitoring.html*

[long92]

Darrell D. E. Long, *A Replicated Monitoring Tool* Workshop on the Management of Replicated Data, 1992, available online at *http://citeseer.nj.nec.com/long92replicated.html*

4.2 Software Tools

Note: Each link is followed by a short description of the package, taken directly from the package's web page.

Multi Router Traffic Grapher (MRTG) http://people.ee.ethz.ch/~oetiker/webtools/mrtg/

The Multi Router Traffic Grapher (MRTG) is a tool to monitor the traffic load on network-links. MRTG generates HTML pages containing GIF images which provide a LIVE visual representation of this traffic. Check http://www.ee.ethz.ch/stats/mrtg/ for an example. MRTG is based on Perl and C and works under UNIX and Windows NT. MRTG is being successfully used on many sites around the net

NetSaint Network Monitor http://www.netsaint.org/

NetSaint is a program that will monitor hosts and services on your network. It has the ability to email or page you when a problem arises and when it gets resolved. NetSaint is written in C and is designed to run under Linux, although it should work under most other *NIX variants. It can run either as a normal process or as a daemon, intermittently running checks on various services that you specify. The actual service checks are performed by external "plugins" which return service information to NetSaint. Several CGI programs are included with NetSaint in order to allow you to view the current service status, history, etc. via a web browser.

Cricket http://cricket.sourceforge.net/

Cricket is a high performance, extremely flexible system for monitoring trends in time-series data. Cricket was expressly developed to help network managers visualize and understand the traffic on their networks, but it can be used all kinds of other jobs, as well.

RR Tools http://cricket.sourceforge.net/rrd.php

RRD is the Acronym for Round Robin Database. RRD is a system to store and display time-series data (i.e. network bandwidth, machine-room temperature, server load average). It stores the data in a very compact way that will not expand over time, and it presents useful graphs by processing the data to enforce a certain data density. It can be used either via simple wrapper scripts (from shell or Perl) or via frontends that poll network devices and put a friendly user interface on it.

NTop http://www.ntop.org/ntop.html

ntop comes with two applications: the 'classical' ntop that sports an embedded web server, and intop (interactive ntop) is basically a network shell based on the ntop engine. intop provides a powerful and flexible interface to the ntop packet sniffer. Since ntop has grown so much in functionality and it cannot be simply considered a network-brower, the problem of capturinag and showing network usage has been split. As of version 1.3 the ntop engine captures packets, performs traffic analysis and information storage. intop implements a bare, command line based interface, with an apparently spartan look and feel, but a lot of functionality already implemented, and others planned for future releases.

A Collection of command-line management tools http://www.ibr.cs.tu-bs.de/projects/scli/

The scli package was written to address the need for small and efficient command line utilities to monitor and configure network devices and host systems. The scli package is based on the SNMP management protocol. It utilizes a MIB compiler called smidump to generate C stub code. In fact, virtually no SNMP knowledge is required in order to extend the scli programs with new features.

Remstats http://silverlock.dgim.crc.ca/remstats/release/index.html

Remstats is a system of programs to: gather data from servers and routers, store and maintain the data for long periods, produce graphs and web-pages tieing them together, and monitor the data for anomalous behaviors and issue alerts. It is built on RRDtool.

MATtool http://www.ee.ryerson.ca:8080/~sblack/mat/

MATtool is an easy to use network enabled UNIX configuration and monitoring tool. It provides an integrated tool for many common system administration tasks, including Backups., and Replication It includes a warning system for potential system problems, and graphing of many common system parameters.

BigSister http://bigsister.graeff.com/

Big Sister is a clone of Sean MacGuire's Big Brother. Its primary functions are: monitor networked systems; provide a simple view of the current network status; generate alarms on status changes; generate a history of status changes; interoperate with other Big Sister or Big Brother instances or foreign network monitors (such as HP Openview).

Snips http://www.netplex-tech.com/software/snips/

SNIPS (System & Network Integrated Polling Software) is a system and network monitoring software that runs on Unix systems and can monitor network and system devices. It is capable of monitoring DNS, NTP, TCP or web ports, host performance, syslogs, radius servers, BGP peers, etc. New monitors can be added easily (via a C or Perl API)

5 Glossary

Availability

The fraction of the time the system is available to service users' requests. The time during which the system is not available is called **downtime**. From the definition, it follows that the availability is a real number between 0 and 1 (inclusive).

Efficiency

The ratio of maximum achievable throughput (stable capacity) to nominal capacity. For example, if the maximum throughput from a 100-Mbps (megabits per second) Local Area Network is only 85 Mbps, its efficiency is 85%.

Reliability

The probability of errors, or the mean time between errors.

Response Time

The interval between user's request and the system response.

Throughput

Rate (requests per unit of time) at which the requests can be serviced by the system.

Utilization

The utilization of a resource is the fraction of time the resource is busy servicing requests.