Monitoring and Management of Large Distributed Computing Systems at FERMILAB

E. Berman, L.L. Carpenter, J. Fromm, K. Genser, L. Giaccheti, T. Jones, T. Levshina, I. Mandrichencko, S. Naymola, D. Petravick, R. Thies, R. Thompson

Fermi National Accelarator Labortory,
Batavia,Illinois, USA

Abstract

The computing needs for future projects at Fermilab will include thousands of computers. This will add to the complexities of operating and managing of a large computing facility. The need for a Distributed Management System (DMS), that is able to efficiently run Fermilab's computing systems, has been recognized and investigations of these types of systems have begun. The comprehensive set of requirements has been defined. Several available products have been evaluated based on the proposed requirements. The proposal to develop proprietary DMS is now under consideration because none of the evaluated products fully satisfied the requirements.

Keywords: distributed management system, network, SNMP, alarm

1 Introduction

The rapid increase in number of computers for Fixed Target experiments and changes for Run II computing add to the complexities of operating and managing of a large computing facility at Fermilab. The Next Generation Operations working group (NGOP) was gathering the requirements for the distributed management system for quite some time. We were looking for pro-active tools that provide self-management of different operatin systems (OS) and mission-critical applications. This paper provides the evaluation and comparison of a few available management systems.

2 Requirements

2.1 Key definitions

We define here basic terminology that will be used throughout the paper:

- `Monitoring Object` is one of the following:
  - `Host` - computer identified by its full domain name
  - `Cluster` - collection of `Hosts`
  - `Component` - atomic element that has a well defined behavior
  - `System` - collection of `Components`
- `Object Tree` - hierarchy of `Monitoring Objects`
- `Condition` - predefined state of `Monitoring Object`
- `Event`- description of detected `condition` that could include `Host`, `Cluster`, `System`, `Severity`, date etc.
- `Action` - activity initiated by DMS depending on the `event`, configuration, current day/time etc. in order to somehow correct the problem.
- `Alarm`- asynchronous indicator generated by DMS upon event reception based on alarm configuration.

- `Monitoring Agent (MA)` - daemon process that is running on a `host` and able to generate `event` based on `condition` and perform predefined `actions`.

2.2 Essential features

The following DMS features have been defined as essential by NGOP . The system should

- be based on industry standards (e.g. SNMP) and provide self-management of different OSs(such as Irix, Aix, HP, Linux, SunOs, and NT) and mission-critical applications.
- be able to detect hardware, network, system and application problems.
- supply basic information such as system daemons status (e.g inetd, syslogd), CPU utilization/load, number of processes and users , runaway processes, network traffic , size of critical file systems, system log errors/warnings (e.g. automount failures), security breaches.
- be scalable up to 1000s of hosts, handle multiple users (e.g. data center personnel, system administrators)
- be secure and support different user authorization levels
- provide the hook for user defined monitoring tools
- generate alarm based on severity of the `event`
- perform ”healing” `actions`
- provide monitoring via Web browser as well as via GUI or command line interface.
- be able to dynamically configure monitoring systems, alarm severity, notification methods
- provide hierarchical view of the entire monitoring system
- restore it configuration across reboots.
- provide qualitative descriptor of ”special” node state such as known bad, off-line, etc...

2.3 Important and desirable features

NGOP outlined a few import and desirable features such as DMS ability to handle overlapping clusters, provide means to generate reports and statistics based on selected criteria, supply contextual on-screen help, implement step-by-step notification regarding performed actions.

3 Evaluated Products

3.1 Why not commercial products?

Based on NGOP analysis of existing commercial software, we came to conclusion that most of the commercial management products would provide limited off-shelf functionality. The substantial efforts and human resources would be required during the installation and customization of the software. In addtion the purchase of third-party products is often necessary in order to gain better scalability via Web integration and more off-shelf functionality [1]. The high initial and support cost of the commercial software was considered as well. Those were the reasons why we decided to focus our efforts on evaluation of free DMS products.

3.2 Freeware Products

We have evaluated several freeware products such as `patrol`[1] [2], `scotty/tkinter`[2] [3], `nocol`[3] [4] and `netlogger`[4] [5]. The evaluation and comparison of these products are listed in the table below:

[1] developed at SLAC/DESY

[2] developed at Technical University of Braunschweig /Network Management Group

[3] developed at Netplex Tecnologies Inc

[4] developed at Lawerence Berkeley NationalLab/Future Technologies Group

Table I: Products Evaluation and Comparison

Products	**Patrol**	**Scotty**	**Nocol**	**Netlogger**
Ported OSs	Irix, HPUX AIX, SunOS, OSF, Linux	SunOs, Ultrix, Irix, Aix,HPUX Solaris, Linux,	SunOs, Solaris, OSF, neXt, Linux, Ultrix	SunOs, Linux, Irix
Language/ Products Dependency	Perl, Java Script	Tcl,Tk, C	C, Perl, curses	C, C++, Java, Perl, Tk, Tcl,flex, bison, pthread
Off-Shelf Functionality	Process,Host, File System Info	SNMP,ICMP, DNS,RPC,NIS, NTP,UDP	SNMP,ICMP,DNS, DNS, RPC, TCP ports Ethernet load,	SNMP,ICMP, Netsat,Vmstat, Iostat
Off-shelf Functionality (Action)	Writes messages to syslog, sends mail, page, kills, restarts process	Writes messages to syslog, periodically invokes predefined jobs.	Invokes predefined job upon event reception.	Can not perform action.
Architecture	patrol is periodically started by cron It collects data defined in configuration file, stores it in log file and computes changes from the previous run. ’Rcps results to the server where html file is generated.	tkinted allows interactively create and maintain object tree. MA, started via tkinetd, runs as a separate process that could communicate with tkinted.	noclogd collects events from MA. It writes events in a standard format suitable for post processing. MA runs on some hosts, pulls data from cluster and sends it to noclogd. netconsle displays information from the events log.	netlogd collects events from MA. It writes events in a stndard format suitable for post processing. MA runs on each monitored hosts and pushes data to netlogd. nlv allows realtime viewing of the events log.
Scalability	Not scalable for big clusters.	Not scalable for multiple users.	Scalable	Scalable
Customization	One level of hierarchy. No overlapping clusters. User could not filter reported events.	Multiple levels of hierarchy. Clusters could overlap. Monitoring tree could be configure via GUI. Events could be flitered.	No notion of hierarchy, clusters,etc. Some events purging is possible.	Possible to group the events by some criteria. No customization is provded for MA.
GUI/UI/ Web Interface	GUI/UI do not exist;WEB interface has limited customization option.	GUI provides a framework for an extensible monitoring; UI exists as well.	UI and “GUI” (curses) allow to monitor events log;Web display is primitive.	GUI allows log file visualization but is unsuitable for data center needs.
API	no	yes(tcl api)	yes(perl api)	yes(c, c++,java, fortran, python)

4 Conclusions

Distributed System monitoring is well recognized as a challenging task. Many commercial as well as open source products try to solve it in many different ways. None of evaluated products fully satisfies our requirements:

- off-shelf functionality (patrol, netlogger)
- scalability (patrol, scotty/tkinter)
- level of customization (nocol,netlogger)
- ability to create hierarchy of monitoring objects (patrol,nocol)
- suitability for the needs of data center personnel (all)

Each product provides some valuable ideas and useful tools:

- existence of logging daemon (nocol, netlogger)
- ULM (Universal Logger Message) as standard logging format (netlogger)
- implementaion of escalating alarms (nocol)
- nlv (netlogger graphical tool) as a events log viewer
- interactive creation of cluster hierarchy (scotty/tkinter)

Based on the evaluation described in this paper we came to the conclusion that acustom DMS should be built at Fermilab in order to:

- meet our own requirements
- have a flexible and maintainable system
- be able to extend in the future as need arises

We have all prerequisites to successfully accomplish this goal due to our in depth understanding of the requirements and level of expertise in technology and tools. The experience gained from analysis of existing systems, and possibility to use some of freeware tools will greatly facilitate our efforts.

References

1 M. Jander, “Framework Fraud?”, Data Communications,9:33-42, September 21, 1999.
2 http://www-d0en.fnal.gov:/patrol
3 http://www.ibr.cs.tu-bs.de/projects/scotty
4 http://www.netplex-tech.com/software/nocol/
5 http://www-didc.lbl.gov/NetLogger