MODIFICATIONS TO CLUSMON:
A BEOWULF CLUSTER MONITORING SYSTEM

by

Madhura Phansalkar

A project

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Computer Science

Boise State University

May 2009

This project presented by Madhura Phansalkar entitled *Modifications to Clusmon: A Beowulf cluster monitoring system* is hereby approved:

Amit Jain Advisor	Date
Teresa Cole Committee Member	Date
Jyh-haw Yeh Committee Member	Date
John R. Pelton Dean of the Graduate College	Date

dedicated to my family

ACKNOWLEDGMENTS

First and foremost I would like to express my deepest gratitude to my advisor, Dr. Amit Jain for his support and guidance in the completion of this project. Dr. Jain has been extremely helpful, very patient and encouraging during my entire graduate study at the Boise State University. I have learnt much from Dr. Jain as my advisor as well as a good friend, and for that I would always remain indebted to him.

I would also like to thank my committee members: Dr. Yeh and Dr. Cole for being on my committee and for taking the time to read my project report. I would like to acknowledge Dr. Uh, Dr. Sirisha Medidi, and Dr. Murali Medidi for their support and encouragement during my entire graduate study.

Most importantly, I would like to express love towards my husband, Hrishi for his support, help, and encouragement during the tough times of my school life. Without his encouragement, I would not have finished graduate study. I would also like to thank all my friends, colleagues for being who they are and keeping me sane. Finally, I would like to express my love and gratitude to my parents, in-laws, grandparents, my sister Anuja and all relatives for their unconditional love, affection and support, without which this would not have been possible.

ABSTRACT

Clusmon is a Beowulf cluster monitoring tool that aids cluster administrators in monitoring the cluster health remotely. Clusmon, with its unique three-tier architecture is intended to be a design alternative. It supervises various parameters on the cluster nodes such as CPU temperature, fan speed, CPU load, memory usage, network traffic, etc and displays a web based dashboard of collected data. It is fast and efficient in that it provides the updates within an interval of ten to fifteen seconds.

In this project Clusmon was mainly studied for scalability and stability. We demonstrated that Clusmon can supervise a cluster with up to 3000 physical machines and provides all the updates within the update interval. Clusmon is highly robust and has very low failure rate. Stability of Clusmon is assured by carefully handling a wide variety of exceptions and failures. By analyzing the code and performing required modifications, Clusmon was observed to be stable, scalable and maintainable. RPM installers were created for Clusmon that automate the long installation process and allow customized installation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Clusmon

Clusmon is a Beowulf cluster monitoring system that assists cluster administrators in monitoring the overall behavior of a cluster. Clusmon supervises nodes on a cluster and gathers cluster data. This data is represented as a dashboard of graphs and dials. Clusmon supports monitoring various features of the cluster such as CPU temperature, system temperature, uptime, fan speed, memory usage, network traffic, CPU load, etc. It is designed to move data analysis to another location, on a machine that is not part of the cluster.

There are other cluster monitors that are available, but Clusmon has a unique design that uses a middle end database with a front-end web interface.

1.2 Characteristics of Clusmon

The main characteristics of Clusmon are:

- Easy to install and use
- Allows remote monitoring through web interface
- Fast, efficient, robust, scalable

- Flags hardware failures, sends a warning email to the administrator if cluster temperature exceeds threshold
- Monitors various features such as system temperature, CPU temperature, uptime, fan speed, memory usage, network traffic, CPU load, jobs running and batch processes
- Performs data collection on cluster, and moves data analysis to another location
- Stores data for historical reference
- Easy to configure

1.3 Prior work done on Clusmon

Previous work has been performed on Clusmon that created a complete cluster monitoring system [1]. In 2004, J. Mazerelli, an undergraduate student, wrote PHP front-end and designed a graphical interface for Clusmon. He also wrote the first back-end version in C. C. Kennington, a graduate student at Boise State University rewrote the back-end in Java as part of his Master's thesis in 2006 [1].

Clusmon was under observation for its stability for a long period of time. As Clusmon is a monitoring system, it is required to be robust and run without any failures. During the observations, Clusmon crashed due to some exceptions in the backend Java daemons. Moreover, Clusmon was never studied from the scalability perspective. Thus problems arising due to scalability were unknown. All such observed and discovered issues required revisiting the back-end daemon design and code.

The task of modifying the back-end and resolving the observed and any newly discovered problems in Clusmon to make it stable and scalable was the main goal of this project.

CHAPTER 2

BACKGROUND

Earlier work on Clusmon created a fairly complete and functional cluster monitoring system [1]. The study was targeted for designing the Clusmon to follow three-tier architecture.

- The back end consists of daemons that collect data. It was developed in Java.
- The middle layer is a database which stores the gathered data.
- The front end is a web interface developed in PHP, which represents the data in the database in a graphical format that is easy to interpret and monitor.

Backend daemons consist of Head and Node daemons. These daemons are background processes that perform data collection and analysis. The head daemon runs on master node. It collects information on individual node of the cluster by querying the node daemon running on each of the compute node. All the data gathered are passed on to web daemon which is the middle layer of Clusmon. It updates the database periodically with the received data. Front-end PHP web interface refers to the database to display dashboard of the cluster data. Following figures show Clusmon front end (Figure 2.1) and its architecture (Figure 2.2) [1].

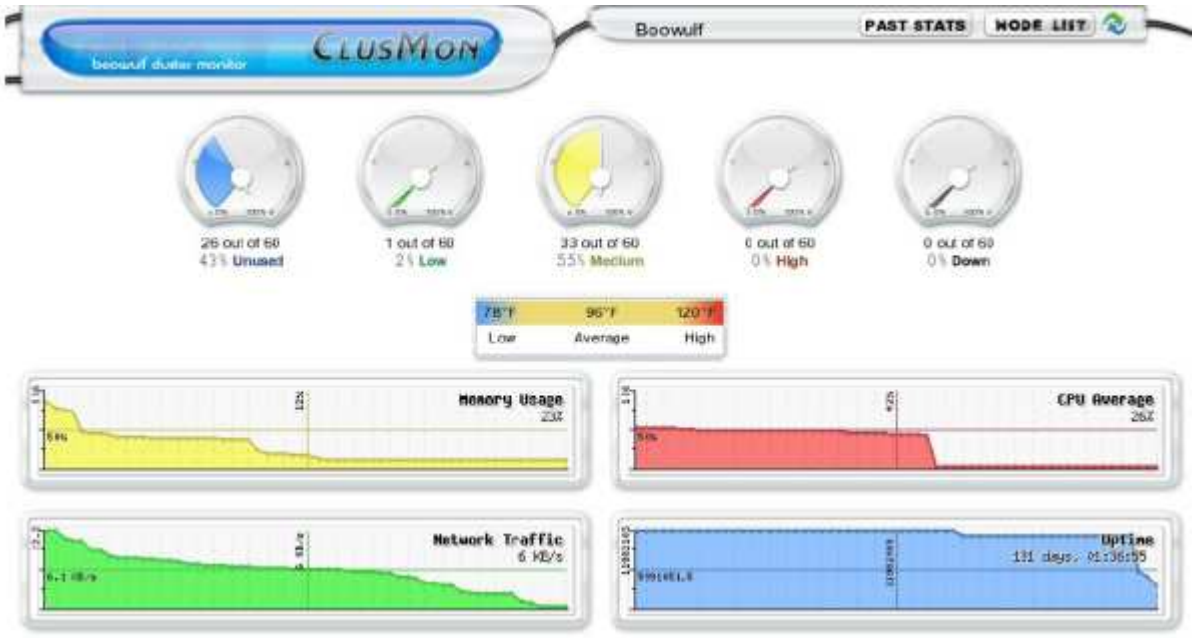

Figure 2.1: Clusmon front end

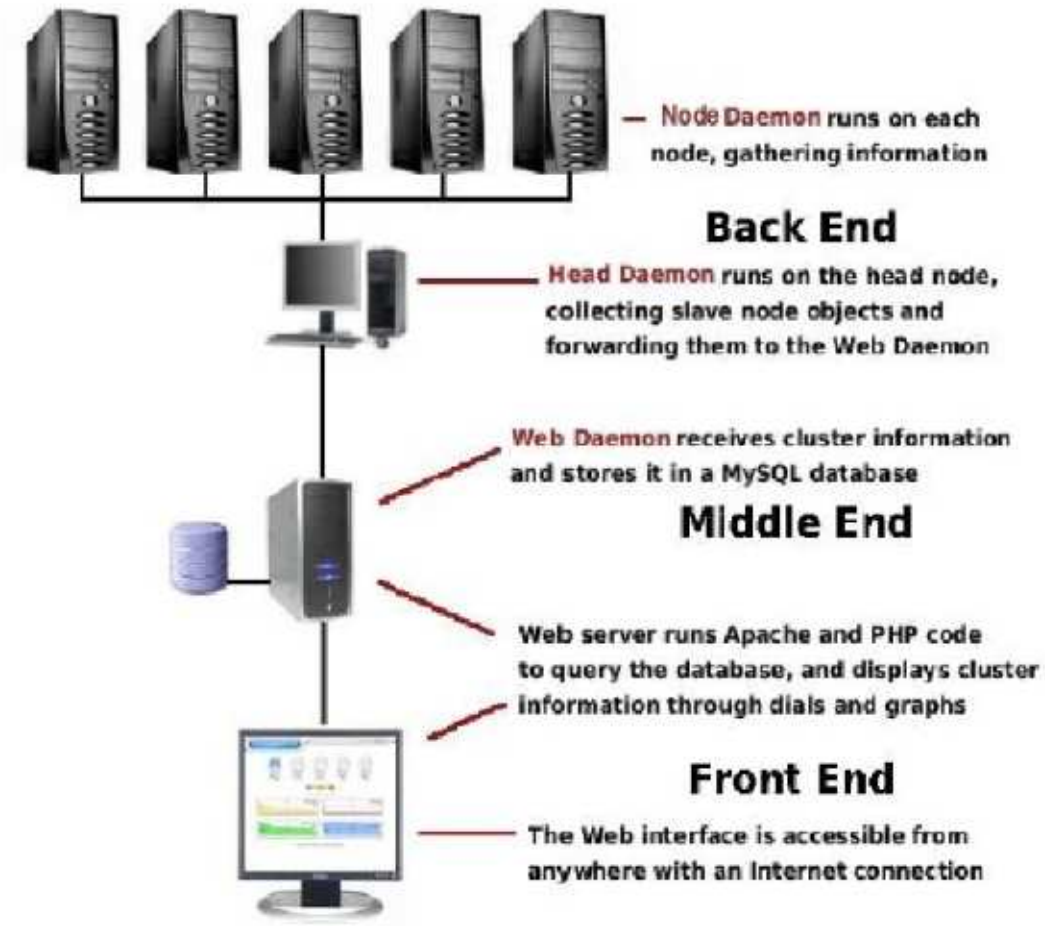

Figure 2.2: Clusmon architecture

There are several other existing solutions, both open-source and commercial. Some of the more popular ones are listed below [1].

Ganglia [2], by the University of California, Berkeley, is one of the most popular and widely used open source cluster monitor. Similar to Clusmon, it uses a multicast for discovering and registering new nodes. Thus both maintain a dynamic list of nodes to monitor. However, one major difference is that Clusmon uses binary data structures for transferring the cluster data, while Ganglia uses XML.

openSSI-webview [3] by Kilian Cavalotti is another open source cluster monitor which provides quick view of overall cluster state including hardware and status information on nodes. It also allows migrating user processes all across the cluster. It uses PHP scripts to display data in graphical format and shell code for gathering data on nodes.

Clusterprobe [4] is an open-source cluster monitor by Z. Liang, Y. Sun, C.Wang, It is developed in Java and uses RMI (remote method invocation) for communication. It differs with Clusmon in that it does not collect temperature data. It works by dividing the cluster into subsections, and running a monitoring proxy per subsection.

ClusterWorX [5], by Linux Networx, is a commercial, all-purpose cluster management tool that collects data at an extremely fast rate. While it is a very robust solution, it is also very expensive. It has the restriction that it does not allow remote monitoring. Unlike Clusmon, an administrator would need to be physically present at the cluster to observe the cluster behavior.

Supermon [6] is an open source cluster monitor developed by the Advanced Computing Laboratory. It only returns changed data thus allowing high sample-rate monitoring. Unlike Clusmon, Supermon uses s-expressions which resemble lists in LISP.

Each solution uses its own approach to gather and store data. However, all monitors compute the data structures locally meaning that they perform the majority of data analysis on the cluster itself. Clusmon stands out with its feature of collecting the data on the cluster, and analyzing it at another location which is not part of the cluster. Clusmon satisfies the intention of being a design alternative.

CHAPTER 3

MODIFICATIONS TO CLUSMON AND SCALABILITY STUDY

The purpose of revisiting and modifying the code was to make the daemons more scalable, stable, readable, and maintainable. Additionally, the aim was to resolve observed and discoverd bugs. The study was initiated with scalability experiments.

3.1 Scalability study

The goal of this study was to scale Clusmon by growing the number of compute nodes, observe the time required to grow the nodes and update the database with the information collected on nodes and to observe Clusmon behavior as it is scaled.

3.1.1 Experimental setup

It was decided to scale the number of nodes up to 3000. It was hard to get a cluster with 3000 physical machines connected to the master node. The Boise State Beowulf cluster with 60 physical machines was available as the test environment. A cluster of 3000 machines was simulated by running multiple copies of node daemon on each of the 60 physical machines of the Boise State Beowulf cluster.

3.1.2 Code modifications required for simulation

For the simulation, we required to run multiple copies of node daemon on a machine acting as multiple physical machines. Node daemon was initially designed to run on a dedicated port which was recognized by head daemon when it attempted to discover the compute node. The port number was read from the configuration file (`clusmon.node.config`). The node was identified by IP address while receiving multi-cast packets from that node. This design allowed only single copy of node daemon to run on a machine.

Node daemon code was modified to accept the port number as a command line argument and bind itself with that port in rmiregistry. An unique identifier was generated for each daemon with a combination of IP address and port number. Thus multiple copies of node daemons, sending multi-cast packets through the same IP address could be treated as different nodes. Head daemon code was modified accordingly to discover node daemons with their corresponding port numbers and IP addresses.

3.1.3 Scalability issues

Several problems were observed within minutes when Clusmon was scaled beyond 1000 nodes. The first thing we noticed was concurrent modifications to the shared data structure in the code. Since Clusmon makes heavy use of multithreading concept, synchronization is the main bottleneck. In another occurrence, Clusmon head daemon died due to "Too many open files" exception. This clearly indicated that the problem occurred while creating sockets. The file-max limit set up by the operating system

(usually 1024) was found to be insufficient for our simulation. Increasing the limit allowed creating required extra socket connections.

The Clusmon web-daemon suffered from a crash due to "Too many open connections" when a large number of threads created a connection to the database. The problem was fixed by closing open connections when database modifications were finished.

This study discovered many new problems and their solutions, thus leading to a more stable Clusmon. Several issues related to synchronization were taken care of by handling race conditions using semaphores.

3.2 Results and analysis

After applying the fixes to the problems Clusmon was observed to scale up to 3000 nodes without any issues. Additionally, stress testing was performed on Clusmon which indicated that Clusmon was able to keep up with 5000 simulated compute nodes. Number of nodes was increased in fixed interval such as 10 nodes, 500 nodes, etc. Total time to grow and the time for database update were captured by instrumenting the code.

The following results were obtained for various experiments performed on Boise State Beowulf cluster. The results indicate average time in milliseconds using 60 physical machines, scaling Clusmon up to 3000 nodes.

3.2.1 Experiment 1

The first set of experiments was performed using an increment of 500 nodes. Following table shows the performance of Clusmon.

Table 3.1: Timings observed for number of nodes with increment of 500

Number of nodes	Time to grow (ms)	Time for updating database (ms)
100	12395	1068
500	1816	2158
1000	3309	3859
1500	42850	10340
3000	301636	23900

Figure 3.1 shows the plot of the "Time to grow" specified in Table 3.1. Figure 3.2 plots the data point for "Time for updating database" in Table 3.1. It is observed that initial time to grow certain number of nodes varied depending on the order of events followed by head daemon. The time taken to discover the nodes and extract information on individual node fluctuated as multiple threads were involved in the process. Moreover, the thread timings varied depending on number of nodes discovered at a particular instance. Thus the data points in Figure 3.1 are observed to be nearly exponential. However, growing the nodes is an infrequent operation.

It is observed from Figure 3.2 that the time taken for database update is nearly linear. Clusmon uses an update interval of ten or fifteen seconds. It is observed that updates were captured within this interval for 1500 compute nodes.

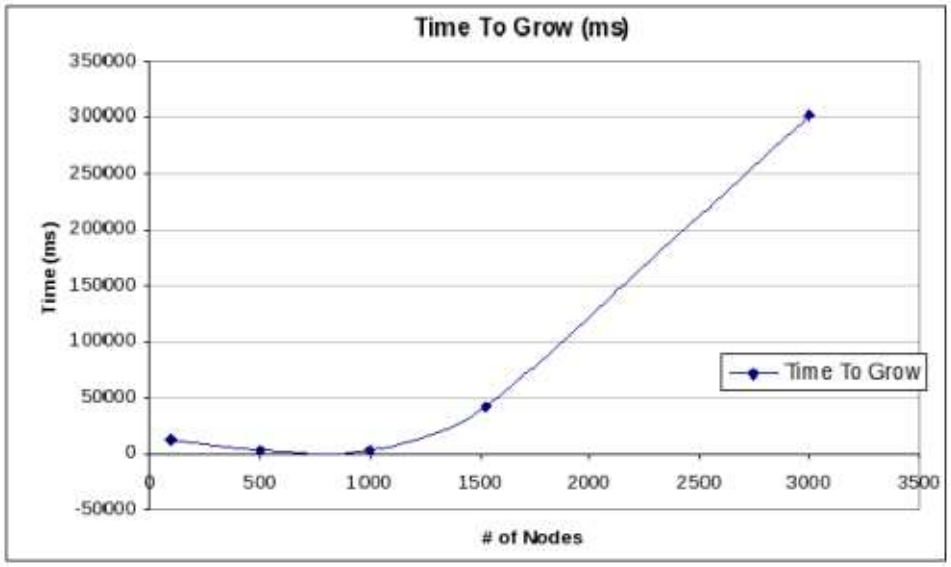

Figure 3.1: Time to grow compute nodes with an increment of 500

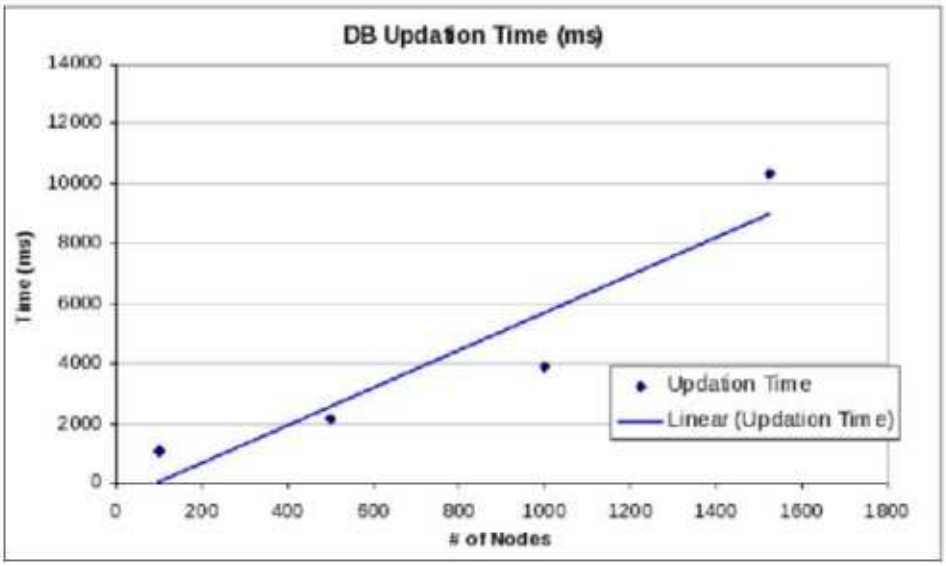

Figure 3.2: Database update time for number of nodes with an increment of 500

3.2.2 Experiment 2

Another experiment was performed to verify the linearity observed in experiment 1. Number of nodes was grown with an increment of 10. Table 3.2 illustrates the results for this experiment. Figure 3.3 plots the data points for database update specified in Table 3.2.

Table 3.2: Timings observed for number of nodes with increment of 10

Number of nodes	Time to grow (ms)	Time for updating database (ms)
10	4605	690
20	1072	812
30	1091	823
40	1074	837

Similar results were obtained from this experiment showing that initial timings to discover and grow nodes may vary. However, time taken for database update is linear. These results support the linear behavior of web daemon to update the database.

Based on the data in Table 3.2, time taken to update the database was extrapolated for 3000 compute nodes. The extrapolated value was observed to be 14237.5 ms. Thus it can be estimated that updates would be available within fifteen seconds for 3000 compute nodes.

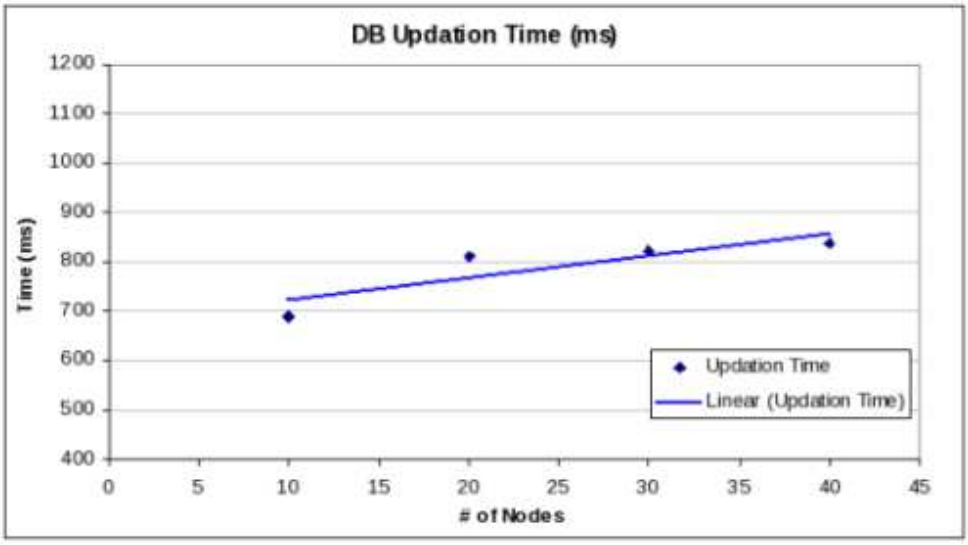

Figure 3.3: Database update time for number of nodes with an increment of 10

3.2.3 Experiment 3

In this experiment, Clusmon was scaled up to 100 and 500 nodes. Instead of incrementing the number of nodes in intervals, all compute nodes were started at the same time. The goal was to observe the Clusmon behavior when all nodes are added at once. Table 3.3 states the timings observed in this experiment.

Table 3.3: Performance of Clusmon for 500 compute nodes

Number of nodes	Time to grow (ms)	Time for updating database (ms)
100	12395	1068
500	65522	2808

Clusmon takes 12.3 seconds for initial discovery and growth of 100 nodes and 65.5 seconds for 500 nodes. This indicates a nearly linear time as 65.5 is nearly five times 12.3.

3.3 Concluding remarks about scalability study

Clusmon was able to keep up with simulated 3000 compute nodes, which indicates that it can support a cluster with 3000 physical machines. The trend in the time taken to grow nodes varies with the order and length of events followed by head daemon. However the time taken to update the database is observed to be linear in terms of number of nodes. Clusmon captures all the updates in 23.9 seconds with the simulated cluster. The extrapolated value (14.2 ms) shows that Clusmon may report the updates within ten to fifteen seconds on a real cluster with 3000 machines.

3.4 Design reconsideration and modifications

3.4.1 Order of backend daemons

Clusmon placed restriction on the start up order of backend daemons. The design of these daemons forced the web daemon to start before head daemon was started. Head daemon would continuously wait for the web daemon to start, which forces the strict order. After observing such conditions, the design of Clusmon web daemon and head daemon was reviewed. These daemons were redesigned such that they could work independent of each other. This made Clusmon more flexible and faster at start up as head demon can start discovering the nodes even if web demon has not started yet.

3.4.2 Maintainability and readability

The code was made more readable by making it modular. Removing redundancy in the code, refactoring the code and reducing the size of critical sections contributed towards readable code. Additionally, modularity contributed to clusmon's maintainability.

3.4.3 Stability

Stability was assured by verifying that all possible exceptions are handled. Backend daemons were observed to crash due to certain exception handlers not being set up. Clusmon encountered a severe crash due to concurrent modifications to a data structure by multiple threads in multiple classes. Head daemon and listener class shared a linked list that is used hold compute node objects. Simultaneous modifications to

this list caused Clusmon to fail regularly. Separating the data structure locally in each class and ensuring synchronization was the solution.

Another problem was related to open database connections within web daemon. Allowing too many open connections without closing them was the cause of failure. Fixing these fundamental issues moved Clusmon towards stability.

3.5 Other modifications

The code was analyzed for variety of exceptions. They typically include null pointer handling, concurrent modifications to data structures, etc. By making the exception handling more robust, code was made more stable. Furthermore, the code was refactored to make it more readable. Hard coded parameters were removed from the code by making them configurable. For example, the port number used by the web daemon for communicating with the head daemon was made configurable by reading it from the configuration file used by the web daemon. Minimizing the size of critical sections improved the efficiency of the code. Thread synchronizations and critical section handling was made more robust. The stability was assured by testing the code for variety of test cases.

3.6 Testing and debugging

Testing Clusmon was a challenging task. Running Clusmon on multiple nodes and reproducing the failures in critical conditions was the challenge.

Clusmon scalability was tested using shell scripts that would scale the number of nodes up to 3000 by running multiple copies of nodes daemons on 60 physical

machines. The code was instrumented to capture the timings for various events that occur while scaling the nodes and updating the database with the current information on nodes.

Clusmon was tested on three clusters which include Boise State Beowulf cluster, Boise State Onyx cluster and Rookery cluster. The biggest challenge was to reproduce the bugs caused by thread timings. Problems such as concurrent modifications were observed rarely. The code was instrumented so that it regularly produced such failures. This assisted in verifying the solutions to the problems. Handling race conditions and synchronization were the most significant solutions as Clusmon makes heavy use of multithreading.

CHAPTER 4

CLUSMON INSTALLER

Various steps need to be followed to set up Clusmon. Clusmon installer automates the complete set up process. Each of the Clusmon daemons is installed by a separate RPM package. This allows customized installation of individual daemons.

Clusmon code uses self contained Java, MySQL, and Apache installation. This eliminates need to change the existing versions if any and also avoids meddling with the existing system setup. The installer takes care of these custom installations.

4.1 Clusmon installation steps common to all three installers

- Create 'Clusmon' user: The installer creates Clusmon user and sets Clusmon home to `/usr/local/Clusmon`.
- Copy required files: Corresponding files for Clusmon daemons are copied to `/usr/local/Clusmon`.
- Configure Clusmon config files: Parameters such as IP address, Host name are derived dynamically on individual machines and are placed in the Clusmon configuration files.
- Start the daemons: corresponding daemons are started as services.

4.2 Clusmon web daemon installer

The web daemon is usually installed on a separate machine that is not part of the cluster. However it can also be installed on the master node that runs the head daemon. The steps followed by web daemon installer are as follows.

- The custom installation for MySQL and Apache are placed under `/opt/bsaci` directory. This directory is created if it does not exist.
- Any existing listeners on ports 81 and 3307 are shut down.
- Apache web server is started on port 81.
- "clusmon_data" database is created with required permissions assigned to Clusmon user.
- Clusmon web configuration file is updated for database credentials.
- MySQL daemon is started on port 3307.
- Clusmon web daemon is started as service.

4.3 Clusmon node and head daemon installers

These installers follow the basic steps specified as common steps. They update corresponding configuration files and start up daemons as services.

4.4 Packaging with RPM

RPM (Redhat Package Manager) is the most popular software package manager for Linux distributions. It allows building packages in two varieties: binary package,

encapsulating the software to be installed, and source package, containing source code to produce binary package. The packages contains an archive of files and meta-data such as helper scripts, file attributes that are used to install and erase the package. The following steps are required to build a RPM package.

1. The most important step is to create an input file, called spec file. This file tells RPM how to build and package the software. The spec file has several tags and sections. Figure 4.1 shows the tags in `clusmonhead.spec` file. The sections are illustrated in Figure 4.2.

    - RPM creates a temporary directory to build the package. "`BuildRoot`" tag can be used to define this directory. Name, Version, Release tags illustrate the version information about the package.
    - "`prep`" section is used to unpack the source code into a temporary directory and apply any patches.
    - "`pre`" section performs any actions that should be taken before initiating the installation process.
    - "`build`" section is used for compiling the code.
    - RPM runs the "`install`" section to install the code into directories on the build machine. It then reads the list of files from the "`files`" section, gathers them up, and creates binary and source RPM files.
    - "`clean`" section is used to remove the temporary build directory.
    - "`post`" section describes the post installation action. In case of the Clusmon head daemon, `setup-head-daemon.sh` and `config-head.sh` are the helper scripts that are run after the installation.

- "`preun`" and "`postun`" sections execute the scripts used for removing/erasing the package.
- Any existing version of the Clusmon head daemon is uninstalled using the script `uninstall-head.sh` before initiating new installation.

2. The next step is to create the archive of all required files including the source code, make files, helper scripts, etc. The "`source`" tag in the spec file indicates the name of this archive.

3. The files generated are now placed in the appropriate directories. The spec file is copied to `/usr/src/redhat/SPECS` while the archive is copied to `/usr/src/redhat/SOURCES` directory.

4. RPM packages are generated using the command "`rpm -ba clusmonhead.spec`". The source RPM package is generated and placed in `/usr/src/redhat/SRPMS` while the binary package is usually placed in `/usr/src/redhat/RPMS/` directory.

```
Name: clusmonhead
Version: 1.0
Release: 1
Summary: A utility to install Clusmon head daemon as service.
Group: Application/Systems
License: None
Packager: Madhura Phansalkar
Vendor:  Boise State university
URL: mailto:amit@cs.boisestate.edu
Source: clusmonhead-1.0.tar.gz
BuildRoot: %{_builddir}/%{name}-1.0
%description
This untars the source tar ball to install head daemon package.
```

Figure 4.1: Clusmonhead.spec file tags

```
%prep

%post
/setup-head-daemon.sh
/config-head.sh
/bin/rm -f /Clusmon-head.tar.bz2
/bin/rm -f /setup-head-daemon.sh
/bin/rm -f /config-head.sh
/bin/rm -f /uninstall-head.sh
/bin/rm -f /clusmonhead.spec

%pre
%build
%install
/bin/cp Clusmon-head.tar.bz2 /
/bin/cp setup-head-daemon.sh /
/bin/cp config-head.sh /
/bin/cp uninstall-head.sh /


%preun
/bin/cp uninstall-head.sh /

%postun
/uninstall-head.sh
/bin/rm -f /uninstall-head.sh

%files
/clusmonhead.spec
/Clusmon-head.tar.bz2
/setup-head-daemon.sh
/config-head.sh
/uninstall-head.sh
```

Figure 4.2: Clusmonhead.spec file sections

CHAPTER 5

CONCLUSIONS

Clusmon code was analyzed for scalability. Various experiments were performed to observe Clusmon behavior with the number of compute nodes increasing in interval. The experiments identified various problems caused due to scalability. These issues were resolved by modifying backend daemons. It was observed that Clusmon can support a cluster with up to 3000 physical machines. It gathers updates within an interval of ten to fifteen seconds.

The analysis and review of backend design assisted in deciding the direction of modifications. This study discovered known and unknown problems that caused Clusmon to crash under heavy loads. These problems were resolved by changing the design and using Java's strong exception handling mechanism. This project created a modular, readable, maintainable and robust backend for Clusmon. Clusmon RPM installers were created that automate the long installation process.

REFERENCES

[1] Conrad Kennington. Clusmon: A Beowulf cluster monitor, M.S. Thesis, Computer science department, Boise State university, fall 2006

[2] Ganglia. http://ganglia.sourceforge.net

[3] openSSI-Webview. http://openssi-webview.sourceforge.net

[4] Clusterprobe. http://www.srg.cs.hku.hk/srg/html/cprobe/readme of clusterprobe.htm

[5] ClusterWorx. http://www.linuxnetworx.com/clusterworx

[6] Supermon. http://supermon.sourceforge.net