

Monitoring Temperature and Fan Speed Using Ganglia and Winbond Chips

Caitie McCaffrey

Office of Science, SULI Program

Stanford University

Stanford Linear Accelerator Center

Palo Alto, California

August 18th 2006

Prepared in partial fulfillment of the requirements of the Office of Science, U.S.

Department of Energy Science Undergraduate Laboratory Internship (SULI) Program

under the direction of Yemi Adesanya in the Scientific Computing and Computing

Services group at Stanford Linear Accelerator Center.

Participant: _____

Signature

Research Advisor: _____

Signature

Table of Contents

Abstract.....	page 3
Introduction.....	page 4
Methods and Materials.....	page 6
Results.....	page 8
Conclusion.....	page 10
Acknowledgements.....	page 11
References.....	page 11
Figure 1.....	page 9
Figure 2.....	page 10

Monitoring Temperature and Fan Speed using Ganglia and Winbond Chips

CAITIE MCCAFFREY (Cornell University, Ithaca NY 14850) YEMI ADESANYA

(Stanford Linear Accelerator Center, Menlo Park, CA, 94025)

Abstract

Effective monitoring is essential to keep a large group of machines, like the ones at Stanford Linear Accelerator Center (SLAC), up and running. SLAC currently uses Ganglia Monitoring System to observe about 2000 machines, analyzing metrics like CPU usage and I/O rate. However, metrics essential to machine hardware health, such as temperature and fan speed, are not being monitored. Many machines have a Winbond w83782d chip which monitors three temperatures, two of which come from dual CPUs, and returns the information when the sensor command is invoked. Ganglia also provides a feature, *gmetric*, that allows the users to monitor their own metrics and incorporate them into the monitoring system. The programming language Perl is chosen to implement a script that invokes the sensors command, extracts the temperature and fan speed information, and calls *gmetric* with the appropriate arguments. Two machines were used to test the script; the two CPUs on each machine run at about 65° Celsius, which is well within the operating temperature range (The maximum safe temperature range is 77° -82° Celsius for the Pentium III processors being used). Installing the script on all machines with a Winbond w83782d chip allows the SLAC Scientific Computing and Computing Services group (SCCS) to better evaluate current cooling methods.

Introduction

The Scientific Computing and Computing Services group (SCCS) provides computing and communications support for the Stanford Linear Accelerator Center (SLAC). SCCS is in charge of providing the computing power for the various projects at SLAC including BaBar, which is studying the asymmetry of matter and anti-matter, the Gamma Ray Large Area Space Telescope project (GLAST), the Large Synoptic Survey Telescope (LSST), and the Kavli Institute for Particle Astrophysics and Cosmology (KIPAC). All of the SLAC laboratory projects require large amounts of computing power and data storage. In order to accommodate the needs of the labs SCCS has an entire floor in building 50 dedicated to housing the machines. The room was built specifically to store the machines, it includes a raised floor and is heavily air conditioned to help keep the machines from overheating. Also in order to accommodate all the groups at SLAC a number of operating systems are used including Solaris, Red Hat Linux, Windows, and Mac OS X.

In order to keep such a large number of machines up and running Ganglia, a distributed monitoring system for high performance computing systems, is used to monitor machine metrics such as the percentage of CPU being used, the amount of memory used, and I/O rate [1]. Ganglia's software periodically polls the monitored machines' metrics in order to update graphs and the round robin database (RRD). Ganglia is designed for use on systems composed of clusters and grids; at SLAC the nodes are organized into clusters by project or by the type of work a machine does. Ganglia is currently used to monitor about 2000 of SLAC's machines.

Ganglia is an excellent tool in a few ways. First it operates on multiple operating systems. It is also scalable, since it does not increase overhead for every machine but increases on the order of number of clusters [1]. Ganglia also uses a RRD, a system used to store and display time series data so that only a fixed amount of disk space is needed but a seemingly infinite amount of data can be stored. Old data is compressed by averaging it with past data, thereby freeing up disk space for new data. The averaging decreases the granularity of the graphs when viewing longer time scales, but allows the user to see machine metrics over long time periods without having to constantly increase disk space. Ganglia also creates web pages containing graphs and machine statistics from the monitored metrics. These measurements can be viewed on a cluster scale or at the machine level, allowing numerous machines to be monitored and providing information about past performance so that should a problem arise, the data can be used to find a solution.

One problem with Ganglia is that the metrics it measures are fairly generic since it is designed to be used on a broad range of hardware and software. Ganglia does not automatically monitor some critical components of hardware health, such as temperature. Monitoring temperature is extremely important because machine hardware is only guaranteed to work correctly within a specific temperature range. Should the machines operate outside this range, the machine behavior can become unpredictable.

In order to compensate for the lack of personalization Ganglia allows the user to write their own code to monitor other metrics. At SLAC, many of the machines have *lm_sensor* chips on the motherboard. *Lm_sensors* are a collection of chips that monitor power supply voltages, fan speeds, and hardware temperatures, and make this

information available to the user. In order to more efficiently monitor the machines, a Perl interface is developed to monitor temperature and fan speed by utilizing the functionality of *lm_sensor* chips

Perl is the language chosen for the task of implementing the temperature monitoring. Perl falls in the gap between low level programming languages (like C) which are fast with unlimited uses, and high level shell programming which is slower, but requires shorter code. [4]. Perl will always be slower than a language like C or C++ because it is interpreted by Perl's interpreter and not compiled into byte code and stored in an executable file. Perl is designed for programs that work with text 90% of the time and do other things 10% of the time. In particular, Perl's extensive regular expressions, and portability across platforms makes it an ideal choice.

Materials & Methods

Ganglia has a hierarchical architecture. Individual machines are at the lowest level running *gmond*, the Ganglia Monitoring daemon. *Gmond* runs in the background polling machine metrics (CPU usage, I/O rate, etc.) at various intervals and sending this metrics to receiving channels specified by a configuration file. [3]. Ganglia was developed at Berkley and is open source.

Gmetric, the Ganglia metric tool, allows users to expand the base of metrics monitored by *gmond*. Users can write scripts to monitor metrics on individual machines then invoke the *gmetric* command passing a configuration file, a metric name, the metric value, the type of value (string, unsigned integer, signed integer, etc), and the units of the

metric. *Gmetric* sends the data to all the receiving channels specified in the configuration file the data travels up the hierarchy to periodically update the RRD [3].

The original script was designed to run on a group of 510 batch machines referred to as “nomas”. The nomas all have a Tyan Thunder LE-T V1.07 motherboard containing two Pentium III processors. The Tyan motherboards all contain an *lm_sensor*-compatible chip, the Winbond w83782d, which monitors power supply voltages, up to three fan speeds, and up to three temperatures [2]. The information monitored by the Winbond chip can be accessed by running the “sensors” command.

A Perl script entitled “temp-fanWatch” was written to monitor temperature and fan speeds. The script works by invoking the “sensors” command and storing the remaining text lines in an array. Each line is checked to see if it contains fan speed information or thermal data. If it does the appropriate *gmetric* call is made. The program then sleeps for a number of minutes and repeats the process.

The program also tracks temperature rate of change from query to query by saving the previous temperature data. The previous temperature data is used to determine the change per sample rate. If any of the temperatures measured are changing at a rate greater than the specified trigger rate (default 1° Celcius) then the sample time will decrease. If the sample time is not decreased and the rate of change on any of the temperatures equals the trigger rate the sample time remains the same. If all of the change rates are less than the trigger rate, the sample time is increased.

The sample time is decreased by a power of the variable “decrement” which is user specified (defaults to 0.9). The power is calculated based on the thermal input with

the greatest change rate per sample time. The new number of minutes is calculated using the following formula:

$$\text{New Sample Time} = \text{decrement} ^ {(\text{change}/\text{trigger}) * \text{current sample time}}.$$

This way the sample time decreases more rapidly when the change rate per sample is larger.

The sample time is increased by the inverse of the “decrement”. The new number of minutes is calculated using the following formula:

$$\text{New Sample Time} = \text{current sample time} / \text{decrement}$$

It does not increase more quickly if the rate of change is smaller. This is to prevent the sample time from growing large quickly just because the rate of change was small or non existent for one sample. If the sample time grew on the same scale that the sample time decreased the script would be more likely to miss data in the event that a sudden change occurred.

Results

The script is performing well. It was tested on two machines noma0449 and noma0450. Figure 1 and Figure 2 show some of the graphs produced by the RRD as a result of running “temp-fanWatch” on the machine noma0449. The script will be installed on all of the nomas machines to achieve a better picture of operating

temperature, but the two CPUs on each of the monitored nomas operate at 64-65 degrees Celsius on a regular basis.

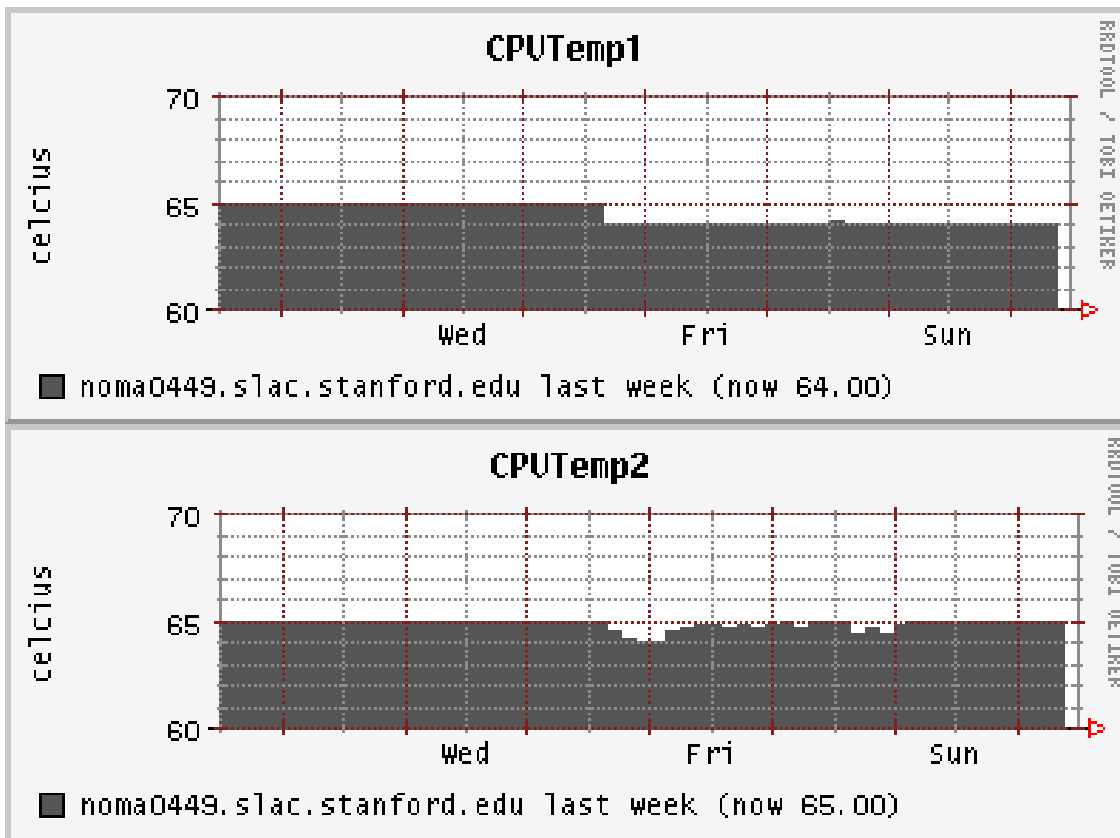


Figure 1. Graphs showing the monitored temperatures of the two Pentium processors on the machine noma0449. The graphs were taken on Monday August 14th 2004 and show the data collected over the previous week.

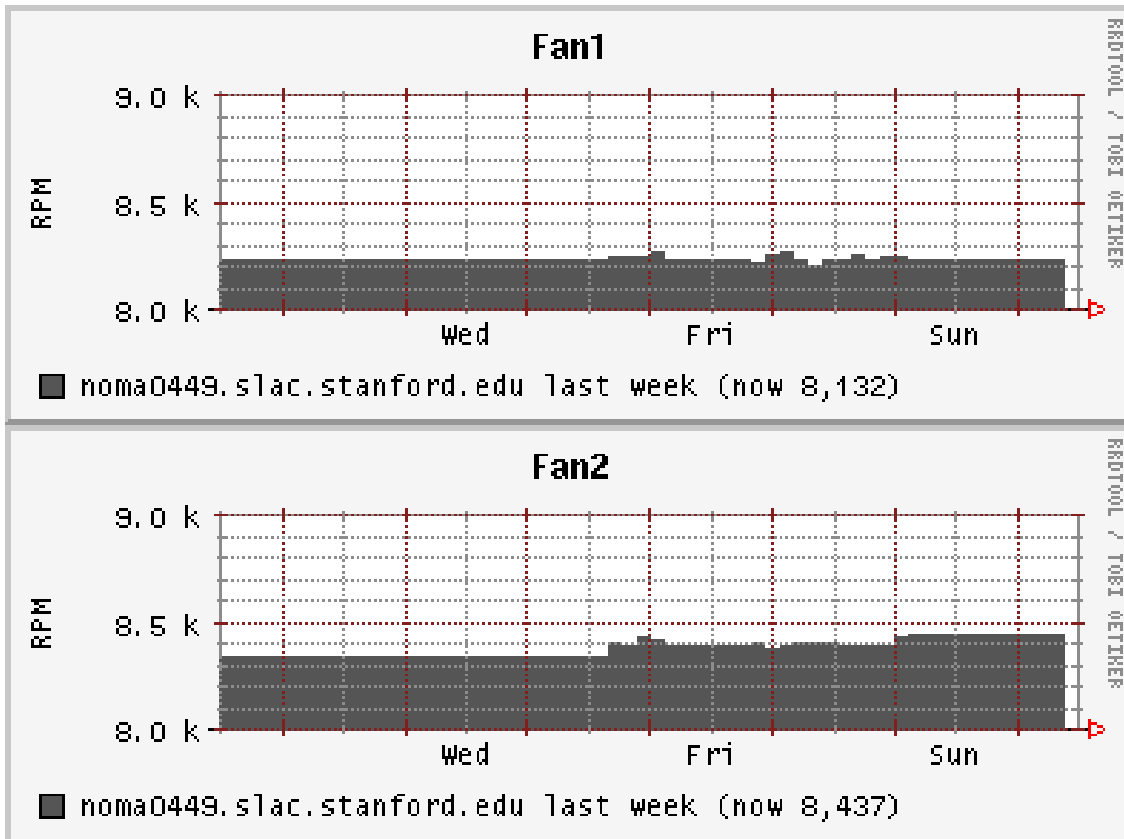


Figure 2. Graphs showing the speeds of the two fans on noma0449's motherboard. The graphs were taken on Monday August 14, 2006 and show the accumulated data over the past week.

Conclusion

The two monitored noma machines used in creating the Perl script are running at acceptable temperatures since the maximum operating temperature range for the Pentium III processors is 77° -82° degrees Celsius. This suggests that present cooling methods are effective; however, the script will need to be installed on the entire noma cluster to fully evaluate present cooling efforts.

Also, although the script was written specifically for the nomas it can be placed on any machine with a similar Winbond chip. The portability of this script will allow

many more machines to be monitored, giving a better understanding of SLAC's machine hardware health.

Acknowledgements

The authors would like to thank the U.S. Department of Energy Office of Science for the SULI internship opportunity. Special thanks to Yemi Adesanya for all his help and guidance throughout the project.

References

- [1] M.L. Massie, B.N. Chun, D.E. Cueller, "The Ganglia Distributed Monitoring System: Design, Implementation, and Experience," *Parallel Computing*, Volume 30, Issue 7, July, pp817-840, 2004
- [2] Winbond Electronics, "W83782D Winbond H/W Monitoring IC", [online document], version 1.1, (2002 October),
Available at <http://www.winbond.com/PDF/sheet/w83782d.pdf>
- [3] "Ganglia README," [online document], version 3.0.0, [2003],
Available at <http://ganglia.sourceforge.net/docs/ganglia.html#authors>
- [4] R.L. Schwartz and T. Phoenix, Learing Perl, 3rd edition, O'Reilly, 2001