Monitoring Cluster on Online Compiler with Ganglia

Nuryani, Andria Arisal, Wiwin Suwarningsih, Taufiq Wirahman
Research Center for Informatics, Indonesian Institute of Sciences
Komp. LIPI, Gd. 20, Lt. 3, Sangkuriang, Bandung, 40135 Indonesia
Email : {nuryani|andria.arisal|wiwin|taufiq}@informatika.lipi.go.id

Abstract

Ganglia is an open source monitoring system for high performance computing (HPC) that collect both a whole cluster and every nodes status and report to the user. We use Ganglia to monitor our spasi.informatika.lipi.go.id (SPASI), a customized-fedora10-based cluster, for our cluster online compiler, CLAW (cluster access through web). Our experience on using Ganglia shows that Ganglia has a capability to view our cluster status and allow us to track them.
Keywords: high performance computing, cluster, cluster monitoring, ganglia

.

1. Introduction

The need for high performance computer which closely related with supercomputer and massively parallel processors (MPP) has been fulfilled with cluster computer. Supercomputer and MPP is so complex to develop and need a high cost while cluster computer can be composed from low cost node computers with high speed network [8]. Therefore, cluster become the de-facto building block for high performance computing [5]. It is made up of nodes, each containing one or more processors, memory that is shared by all of the processors in (and only in) the node, and additional peripheral devices (suck as disks), connected by a network that allows data to move between the nodes [1]. Since clusters today consist of hundreds or thousands of nodes with various hardware and software bases, monitoring system has become another key issue to be addressed.

Monitoring systems reports some set of measurements to higher-level services, therefore, the operator or administrator can monitor the state, operation, and health of all system elements[6,7]. Monitoring system is built for visualizing and monitoring the cluster resource such as memory usage, disk usage, network statistic, number of running processes, CPU load, and other CPU metrics. In addition, analyzing CPU metrics is one way to improve cluster performance. A number of cluster monitoring tool are available, e.g., Ganglia, VAMPIR, DIMEMAS, CACTI, Visuel, Nagios, Zenoss, Zabbix, Performance Co-Pilot (PCP), and Clumon. Each tools has its own strength and weaknesses in certain part compared with the others. For the example, Ganglia is more concerned with metrics gathering and tracking them over time while Nagios has focused in an alerting mechanism, Visuel has an additional function to visualize the performance data of MPI application and Clumon offers web-based pages to show the detailed information about resources, queues, jobs and warnings[2, 3, 6].

At present, we are developing online compiler for high performance computing environment focusing on developing web-interface to ease user (especially programmer) on using cluster infrastructure named CLAW (cluster access through web). Online compiler is a web-based application that provides service for writing, compilation, and execution of source code through web. The CLAW has three parts, i.e., programming interface, monitoring and administration. In this paper we focus on monitoring part only. Two other parts will be presented on other papers.

We use Ganglia to monitor our clusters because of its simplicity to be installed and configured, its extensive graphing capabilities and its web-based interface that allows both current and archival data to be displayed for a whole cluster and for individual nodes. Moreover, Ganglia is easy to be extended to measure and distribute other metrics, using simple command line [6].

This paper is organized as follow. In Section 2, we present an overview of Ganglia and the Installation & Configuration of Ganglia in Section 3. In Section 4 we present our experience on using Ganglia to monitor our cluster and in Section 5 we conclude the paper.

2. Ganglia

Ganglia is an open source distributed monitoring system for high performance computing (HPC) systems such as clusters & grids. It uses technologies such as XML for data representation, XDR for compact, portable data transport, and RRDtool for data storage and visualization. RRDtool is a database designed to store time series of data, e.g., system log at a constant time interval. Ganglia has been used to link and handle hundreds clusters with thousands nodes.

Ganglia view the CPU utilization, i.e., memory usage, disk usage, network statistic, number of running processes, and all other Ganglia metrics. The Ganglia system is comprised of Ganglia Monitoring PHP-based web fronted and two small utility programs, i.e., gmond and gmetad. The architecture of Ganglia is figured in Figure 1.

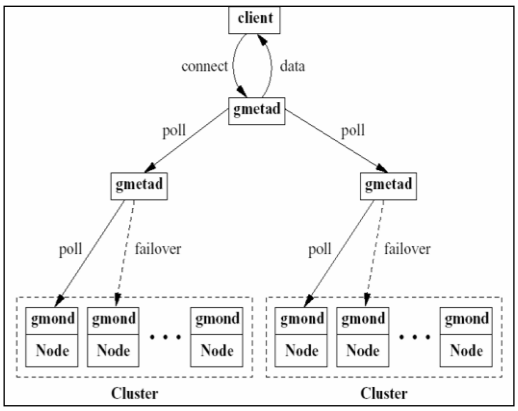

Figure 1. The architecture of Ganglia

2.1 Ganglia Monitoring Daemon (gmond)

Gmond is a multi-threaded daemon which runs on each cluster node being monitored. Gmond is responsible for monitoring changes in host state, multicast relevant changes, listen to the state of all other ganglia nodes via a multicast channel and answer requests for an XML description of the cluster state. Each gmond transmits information in two ways, i.e, unicasting/multicasting host state in external data representation (XDR) format using UDP message or sending XML over TCP connection.

2.2 Ganglia Meta Daemon (gmetad)

Gmetad is responsible for collecting states of the data source (each nodes in a cluster or multiple clusters). Data sources may be either gmond daemons, representing specific clusters, or other gmetad daemons, representing sets of clusters.

Ganglia is using a tree of point to point connections amongst representative cluster nodes to aggregate the state of multiple clusters. At each node in the tree, a gmetad periodically polls a collection of data sources, parses the collected XML, saves all numeric, volatile metrics to round-robin databases and exports the aggregated XML over a TCP sockets to clients. Since each cluster node contains a complete copy of its cluster's monitoring data, each leaf node logically represents a distinct cluster while each non-leaf node logically represents a set of clusters.

2.3 Ganglia PHP Web Frontend

Ganglia-web provides a view of the collection information of each node in a cluster or multiple cluster via real-time dynamic web pages in graphs. The Ganglia web frontend is written in the PHP scripting language, and uses graphs generated by gmetad to display history information.

3. Installation And Configuration

3.1 Installation

As mentioned on the previous chapter, ganglia consists of three components; gmond, gmetad, and web-frontend. There are three installation procedures based on installation packages, which will be described on the next sub-sections.

3.1.1 Source packages

Download the latest source version of Ganglia (extention .tar.gz) from SourceForge website. Execute the following sequence of commands as normal user:

```
$ tar xzvf ganglia-3.1.1.tar.gz
```

Apply node configuration on the installation package:

```
$ cd ganglia-3.1.1
$ ./configure
$ make
```

On the head node, which will give monitoring service with web-front end, the installation configuration must has –with-gmetad option which means that the installation will also configure gmetad and web-front end.

```
$ ./configure –with-gmetad
```

This options requires rrdtool which is by default located in:

```
/usr/include/rrd.h
/usr/lib/librrd*
```

when they located on different directory, the configuration must define their location:

```
$ ./configure \
   CFLAGS="-I/rrd/header/path" \
   CPPFLAGS="-I/rrd/header/path" \
   LDFLAGS="-L/rrd/library/path" \
   --with-gmetad
```

when using gmetad, user has to make sure that they have a directory owned by “ganglia” named

```
/var/lib/ganglia/rrds
```

Install the configured installation package from root account by executing:

```
# make install
```

After every process finished and succeed, ganglia daemons (gmond and gmetad) can be run with

```
# /etc/init.d/gmond start
# /etc/init.d/gmetad start
```

3.1.2 Linux distribution installation packages

For specific Linux distribution, there are specific installation packages. Two most popular Linux distributions which give ease of use for users, Fedora core and Ubuntu has their own installation packages format.

- Fedora core and its derivatives.

  They have a rpm (RedHat package manager). In order to install ganglia from rpm, user has to

download the lastest rpm packages from ther favorite rpm provider (such as http://rpmfind.net/).
Installation can take place by executing:

```
# rpm –ivh     ganglia-gmond-3.1.1-
3.fc10.i586.rpm
# rpm –ivh     ganglia-gmetad-3.1.1-
3.fc10.i586.rpm
```

- Debian and its derivatives (such as Ubuntu).
  They have a deb (Debian package manager). In order to install ganglia from deb, user has to download the lastest rpm packages from ther favorite debian package provider (such as http://packages.debian.org/).
  Installation can take place by executing:

```
# dpkg -i      ganglia-monitor_2.5.7-
5_i386.deb
```

- After every process finished and succeed, ganglia daemons (gmond and gmetad) can be run with

```
# /etc/init.d/gmond start
# /etc/init.d/gmetad start
```

3.1.3 Linux software package management

For specific Linux distribution and system which is connected to online/offline repositories, there are software package management tools which can easily install packages and their dependencies.

- Fedora core and its derivatives.
  They have a yum tool. In order to install ganglia from repository, user has to set up their repository configuration so that if can access to Fedora core repository sites.
  Installation can take place by executing:

```
# yum install ganglia-gmond
# yum install ganglia-gmetad
```

- Debian and its derivatives (such as Ubuntu).
  They have a apt tool. In order to install ganglia from repository, user has to set up their repository configuration so that if can access to Debian repository sites.
  Installation can take place by executing:

```
# apt-get install ganglia-monitor
# apt-get install gmetad
```

- After every process finished and succeed, ganglia daemons (gmond and gmetad) can be run with

```
# /etc/init.d/gmond start
# /etc/init.d/gmetad start
```

3.2 Configuration

3.2.1 Gmond Configuration

Default gmond configuration file is in `/etc/ganglia/gmond.conf`.

The `global` section controls general characteristics of gmond. The attributes of `global` section are `daemonize` to specifiy whether gmond will daemonize or foreground, `setuid` to specify whether gmond will set its effective UID to the uid of the user specified by user attribute, `user` to specify the user, `debug_level` specifies whether gmond will run normally or foreground and outputting debugging information, `mute` to specify whether gmond will send data regardless of any other configuration directives, `deaf` to specify whether gmond will receive data regardless of any other configuration directives, `host_dmax` to specify the maximum amount of time (in second) to flush a host after it has not heard, `cleanup_threshold` to specify the minimum amount of time before gmond will cleanup any hosts or metrics, `gexec` to specify whether gmond will announce the hosts availability to run gexec jobs, `send_metadata_interval` to establish an interval in which gmond will send or resend the metadata packets that describe each enabled gmetric and an optional attribute `module_dir` for indicationg the directory where DSO modules are to be located.

The attribute of the cluster is defined in a `cluster` section. The cluster attributes are `name`, `owner`, `latlong` and `url`. The `name` attributes specifies the name of the cluster, the `owner` specifies the cluster administrator, `latlong` specifies the latitude and longitude GPS coordinates of the cluster and `url` for more information on the cluster. The `host` section specifies of the host, like the location.

The `module` section contains parameters that are necessary to load a metric module. The `module` contains `name` specifies the module's name, `path` specifies the path from which gmond is expected to load the module, `language` (optional) specifies the source code language in which the module was implemented (example : C/C++, Python), and `params` (optional) can be used to pass a single string parameter directly to the module initialization function (C/C++ module only).

The `collection_group` section collects the CPU metrics status and collection time. The `collection_group` section attributes are `collect_once` specifies whether CPU metric will be collected once at startup or not (value : bolean), `collect_every` specifies how many seconds CPU metrics will be collected (value : number) and `time_threshold` specifies the maximum amount of times that can pass before gmond sends all metrics. The `collection_group` section is followed by `metric` section that contain `name`, `value_threshold`, and `title`.

3.2.2 Gmetad Configuration

Default gmetad configuration file is in `/etc/ganglia/gmetad.conf`.

Data source, can be cluster or a grid (comes from another gmetad) to be monitored is defined in the `data_source` section. Format of the `data_source` section is:

```
data_source "my cluster" [polling
interval] address1:port
addreses2:port ...
```

where "`my cluster`" is the name of the cluster to be monitored, `address1:port` is IP address or name and port pair of machine to identity data source. Multiple IP addresses is used for fail-over.

3.2.3 PHP Web Frontend Configuration

Ganglia web frontend configuration file is in `/usr/share/ganglia/conf.php`.This configuration file defines the template, gmetad location, RRDtool location, time range and metrics for graphs.

The `template_name` variable defines the template (like skin) used in the ganglia web. The default and user-defined templates are stored in the "./templates" directory. The gmetad location is defined in `gmetad_root` variable. The `graphdir` variable defines the location for modular graph files.

4. Experience On Spasi

In this section we will present our experience on using Ganglia for monitoring our cluster. We will describe our cluster platform first followed by experimental setup, result and discussion.

4.1 SPASI in a brief

The cluster we have developed, *spasi.informatika.lipi.go.id* (SPASI), consists of 3 linux clusters. The first cluster, *Borobudur*, consists of 6 nodes, each with Pentium !!! 800MHz with 256 MB RAM and fast Ethernet networking. The second cluster, *Kuta*, consists of 4 nodes, each with dual core 2.4GHz Pentium processor with 1 GB RAM and gigabit Ethernet networking. And the third cluster, *Toba*, consists of 8 nodes, each with Intel Core 2 Duo E7400 2.8GHz Pentium processor with 1 GB RAM and gigabit Ethernet networking. Both clusters run a customized fedora10, with kernel version 2.6.27.9-159.fc10.i686 and NFS for file sharing. The daemons were compiled with gcc 4.3.2 and glibc 2.9. The architecture of SPASI, is figured in Figure 2.

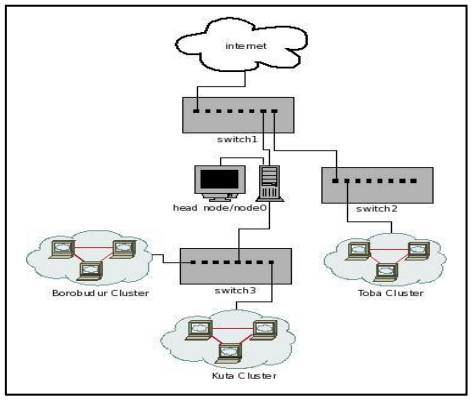

Figure 2. The architecture of SPASI

User access the cluster (head/node0) through internet. Head/node0 has 2 ethernet devices, where *eth0* is the private IP address for cluster network (Private IP class B) and eth1 is the public IP address of our system. In head/node0, also located webserver for our online compiler, *CLAW* (*cluster access through web*). The architecture for *CLAW* is figured in Figure 3 and the homepage of *CLAW* is figured in Figure 4.

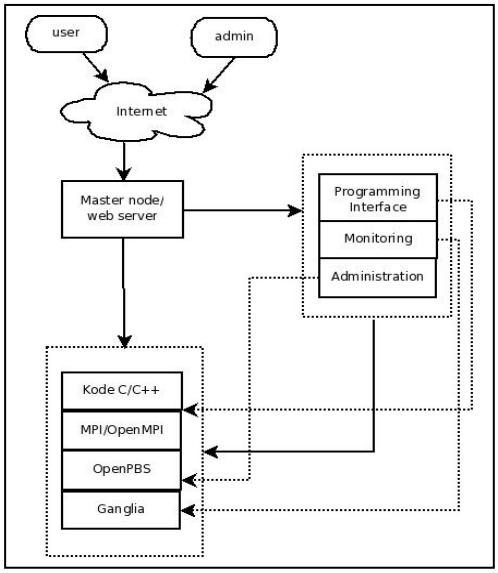

Figure 3. The architecture of *CLAW*

The *CLAW* has three main parts, i.e., programming interface, monitoring and administration. The programming interface menu allows user to write, upload, download, save, compile, execute and also display output and error message. The monitoring menu allows user to monitor a whole of cluster, including resources and networks while administration menu is built for resources management and administration.

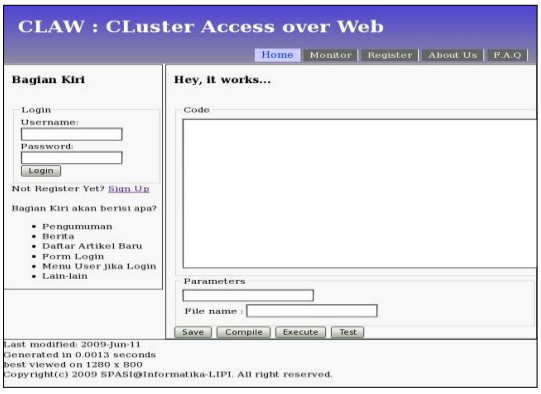

Figure 4. The homepage of CLAW

The *CLAW* has the following specifications: C/C++ code for parallel application code, OpenMPI for parallel programming library and environment, OpenPBS/Torque for resources management and administration, and Ganglia for system and network monitoring.

4.2 Results and Discussion

Monitoring output of ganglia can be access through accessing the ganglia web-front end on the head node. The captured monitoring status screen of *SPASI* are figured in Figure 5 to Figure 7.

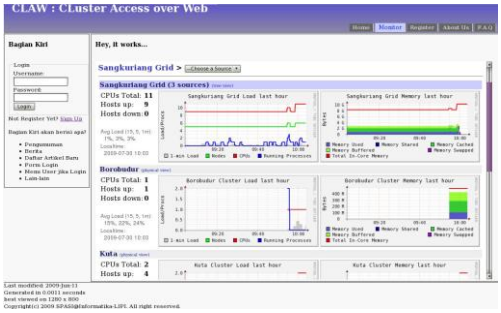

Figure 5. The *spasi* home screen

Figure 6. The *Kuta* cluster physical view

Figure 7. The *Kuta* cluster load view

Based on our experience in using ganglia to monitor our cluster, we found that ganglia reports important and useful node information, such as load (load/processor), memory (bytes), CPU (percent), network (bytes/sec), and packets (packets/sec).

Ganglia reports detailed physical information on every nodes includes CPU count, CPU speed, memory total, last boot time, machine type, operating system, operating system release, location and additional information on swap space total. CPU status reported by Ganglia includes CPU user, CPU system, CPU idle, CPU aidle, CPU wio, CPU intr, and CPU sintr. On memory status Ganglia reports free memory, shared memory, memory buffers, cached memory and free swap memory. On packet status Ganglia reports packets sent, packets received, bytes sent, and bytes received. Disk status reported by Ganglia include total disk space, disk space available and maximum disk space used. Moreover Ganglia is also reports total running processes and total processes.

Ganglia also allows us to track the clusters status in the last hour, day, week, month, and year. Beside that, in our experience on using Ganglia in our fedora operating system-based cluster, Ganglia is very easy to install and configure.

5. Conclusion

Ganglia is a cluster monitoring tool that has a capability to collect and view both the whole cluster and every nodes status in graphs through dynamic web pages. However, Ganglia doesn't have a capability to send some alert mechanism if any problem happened, like provided by Nagios, and also doesn't report on MPI parallel applications execution like in Visuel. As a future work, we will analyze possibilities to customize Ganglia or combine Ganglia with other open source cluster monitoring tools to improve the performance on our cluster monitoring tool.

References

[1] Gropp W., Lusk E, and Sterling T, Beowulf Cluster Computing with Linux, The MIT Press, 2003.

[2] Li KC, Chang HC, The Design and Implementation of Visuel Performance Monitoring and Analysis Toolkit for Cluster and Grid Environments. Springer Science+Business Media, LLC, 2007.

[3] Benincosa V, Ganglia and Nagios, Part 1 : Monitor Enterprise Cluster with Ganglia : Install, Configure, and Extend Open Source Ganglia to Effectively Monitor a Data Center. IBM, 2009.

[4] Benincosa V, Ganglia and Nagios, Part 2 : Monitor Enterprise Cluster with Nagios : Install Nagios to Effectively Monitor a Data Center; Make Ganglia and Nagios work Together, IBM, 2009.

[5] M.L.Massie, B.N.Chun, D.E.Culler, The ganglia distributed monitoring system : design, implementation, and experience, Parallel Computing 30 : 817-840, 2004.

[6] M.S. Jennifer, C. Ben, Monitoring clusters and grids, ClusterWorld, Volume 2 No 7.

[7] Baker, M., Cluster Computing White Paper, University of Portsmouth, UK, 2000.

[8] “High Performance System Laboratory” Research Center, KAIST Computer Science webpage : http://camars.kaist.ac.kr/~nrl/e_index.html
[9] D.S. Federico, J.K. Mason, L.M. Matthew, E.C. David, Wide Area Cluster Monitoring with Ganglia, San Diego Supercomputing Center & Univ of California, Berkeley, 2003.