Ganglia Cluster Toolkit

Brent Chun

Matt Massie

Ganglia (http://ganglia.sourceforge.net/) provides a complete real-time monitoring and execution environment that is in use by hundreds of universities, private and government laboratories and commercial cluster implementors around the world. Ganglia is as simple to install and use on a 16-node cluster as it is to use on a 512-node cluster as has been proven by its use on multiple 500+ node clusters.

Component	Description	Author	Version
authd	obtains and verifies user credentials using RSA public key cryptography	Brent Chun	0.2.0
GEXEC	cluster remote execution system	Brent Chun	0.3.3
Monitoring Core	the daemons and utilities necessary to monitor large clusters in real-time	Matt Massie	2.3.1b4
Web PHP/RRD App	a web interface for viewing (http://ganglia.mrcluster.org/) data from the monitoring core	Matt Massie	1.0.4
Python Class/Client	a Python class for sorting and classifying large clusters using the monitoring core	Greg Bruno	2.0

Ganglia grew out of UC Berkeley Computer Science clustering research: The Millennium Project (http://www.millennium.berkeley.edu/) and it’s predecessor The NOW Project (http://now.cs.berkeley.edu/).

Table of Contents

1. Introduction

Ganglia (http://ganglia.sourceforge.net/) provides a complete real-time monitoring and execution environment that is in use by hundreds of universities, private and government laboratories and commercial cluster implementors around the world. Ganglia is as simple to install and use on a 16-node cluster as it is to use on a 512-node cluster as has been proven by its use on multiple 500+ node clusters.

Ganglia was developed at the University of California, Berkeley Computer Science Division as way to link clusters across the Berkeley campus together in a logical way. Since it was developed at a university, it is completely open-source and has no proprietary components. All data is exchanged in well-defined XML and XDR to ensure maximum extensibility and portability.

Ganglia is compromised of separate components that can work alone or together. The execution components (authd, gexec, and pcp) can run stand-alone or plugged into the monitoring core. The execution system provides high-performance tools for executing programs or manipulating files on your cluster. While, the execution core runs only on Linux at this time, it has been written to be easily ported to other operating systems.

The monitoring core (gmond, gstat, gmetric) allows you to monitor any number of host metrics in real-time. At present, the monitoring core runs on Linux, FreeBSD, Solaris, AIX, and IRIX.

Ganglia is not just a way to link nodes in a cluster together in a logical way but also a way to link clusters to other clusters. Ganglia blurs the line between clustering and distributed computing by providing for C2C (Cluster to Cluster) data exchanges which link disparate cluster resources together into a single logical framework.

2. Ganglia Components

2.1. Components of the Ganglia Execution Environment

The *official* web pages for authd and GEXEC can be found at http://www.cs.berkeley.edu/~bnc/authd (http://www.cs.berkeley.edu/~bnc/authd/) and http://www.cs.berkeley.edu/~bnc/gexec (http://www.cs.berkeley.edu/~bnc/gexec/)

2.1.1. The authd system (authentication daemon)

From the authd author, Brent Chun

“Authd is a software package for obtaining and verifying user credentials which contain cryptographic signatures based on RSA public key cryptography. It includes (i) a server (authd) for authenticating local users through Unix domain sockets and process credentials and (ii) a client library (libauth.a) for requesting new credentials and verifying credentials signed by the server. In the context of clusters, authd is typically used by installing a single cluster-wide RSA public/private key pair on all nodes and running authd everywhere. Given this arrangement, client programs running on any node can obtain and present timestamped credentials to cluster services which can then verify user identities using the cluster-wide public key.

Compared to other approaches for authentication, authd's scheme is attractive since it obviates the need for users to manage their own public/private key pairs."

2.1.2. The Ganglia Execution System (GEXEC)

From the GEXEC author, Brent Chun

"GEXEC is a scalable cluster remote execution system which provides fast, RSA authenticated remote execution of parallel and distributed jobs. It provides transparent forwarding of stdin, stdout, stderr, and signals to and from remote processes, provides local environment propagation, and is designed to be robust and to scale to systems over 1000 nodes. Internally, GEXEC operates by building an n-ary tree of TCP sockets and threads between gexec daemons and propagating control information up and down the tree. By using hierarchical control, GEXEC distributes both the work and resource usage associated with massive amounts of parallelism across multiple nodes, thereby eliminating problems associated with single node resource limits (e.g., limits on the number of file descriptors on front-end nodes). It consists of a daemon, a client program, and a library which provides programmatic interface to the GEXEC system."

2.2. Components of the Ganglia Monitoring Core

2.2.1. The Ganglia MONitoring Daemon (gmond)

The real workhorse of the ganglia monitoring system is the Ganglia MONitoring Daemon (gmond). Understanding how gmond works on each node in your cluster is critical to seeing the "big picture" of how ganglia works. Gmond is simply a single multi-threaded daemon which runs on each cluster node as user "nobody". Installation is as easy as installing a single RPM.

You don't have to have a common NFS filesystem or database, install special accounts, maintain configuration files or other annoying hassles. Gmond is it's own redundant, distributed database as you'll see!

2.2.1.1. A Simple Introduction to Gmond

2.2.1.1.1. Gmond is a Good Listener and Polite Conversationalist

The gmonds running on each node have four main responsibilities: *monitor* changes in the host state, *multicast* relevant changes, *listen* to the state of all other ganglia nodes and *answer* all requests for an XML description of the cluster state. Multicast networking allows a single gmond to simultaneously send information to all remote gmonds.

Gmond as a Good Listener

By default, gmond has two threads which listen to the multicast channel and write the data to a fast, in-memory hash table. All metric data for each cluster node is processed and saved. You might think this would require large amounts of memory but a gmond can maintain the in-memory data in just 16+ 136*n_nodes +364*n_metrics bytes. For example, if you have a massive *1024* node cluster and you are monitoring *25 metrics* on each machine then gmond will only use `16 + 136*1024 + 364*25 = 148380 bytes` or just 144 kilobytes of memory! The hash table is also completely multi-threaded with read/write locks implemented by POSIX thread mutexes and condition variables for each individual host in the cluster. That means that multiple threads can simultaneously read and write to the hash without interfering with each other. Very scalable. I know of a few 500+ node clusters running ganglia.

Gmond as a Polite Conversationalist

Each gmond transmits in two different ways: multicasting host state in external data representation (XDR) or sending XML over a TCP connection.

Gmond only sends updates of the metrics that it is monitoring for two reasons: a change in the value of the metric exceeds the value threshold or when gmond hasn't multicast the metric longer than the time threshold. The value threshold ensures that gmond only multicast when it really needs to. In short, why should it talk if it has nothing new to say? Of course things are always changing and so the time threshold ensures that small changes haven't built into large changes to slowly we don't notice. The time threshold also acts as a heartbeat measure. If a node doesn't multicast in more than the heartbeat interval then the other nodes are alerted that it is likely down. These thresholds also dramatically reduces chatter on the multicast channel. For example, the number of CPUs on a host is only sent once an hour since it is a constant whereas 1-minute load could be sent every minute depending on the change in value.

2.2.1.1.2. Gmond Loves to Meet New Gmonds and Never Leaves a Fallen Comrad

Since all nodes in your cluster are storing your cluster state, it is important that all nodes have the same cluster image. The gmond self-organizes and knows how to make sure that all gmonds are on the same page.

Gmond Loves to Meet New Gmonds

Anytime that gmond gets a multicast packet from a new host, it expires it's time threshold of all its metrics. This means that all its data will get sent to the remote gmond even if it wasn't previously scheduled to be sent. For example, the number of CPUs is only multicast once an hour. If one gmond did not expire its time threshold then the remote gmond wouldn't know the number of CPUs on that machine for up to an hour.

Gmond Never Leaves a Fallen Comrad

Gmond is also good at handling failures. It's inevitable that nodes in your cluster will go down occasionally and (heaven-forbit!) gmond may even fail. Recovering gracefully from those temporary failures is important. If a gmond hasn't heard from a remote comrad during the heartbeat interval and then suddenly starts hearing from that remote gmond again, it will expire the time threshold on all its metrics as if to say to it's fallen comrad, "Welcome back. Here is all I know about myself".

Gmond is Self-Aware

Each gmond processes its own multicast data locally via loopback. That means that it saves in-memory the very data that it's sending to remote gmonds.

2.2.1.1.3. Gmond Doesn't Talk with Strangers

When asked, gmond will write out the complete cluster state in XML including a DTD. However, gmond will only share that data with hosts in its in-memory cluster hash OR a host specified with the trusted_host commandline option at gmond startup. The trusted_host option is a very powerful tool. Say you have two clusters: Cluster A is in California and Cluster B is in Nevada. Setting up multicast enabled routes between your two clusters would likely be a nightmare. However, using the trusted_host option only requires unicast routes between the two clusters. On Cluster A you would start a gmond with

```
gmond --trusted_host xxx.xxx.xxx.xxx
```

and fill in the x's with the IP address of a remote machine on Cluster B. On Cluster B vice versa, start gmond with the trusted_host set to a machine on Cluster A. Consider it a handshake between strangers to build a trust relationship.

If you want to see just how easy it is to get XML description of your cluster, run

```
telnet localhost 8649
```

from machine running gmond. You can also run

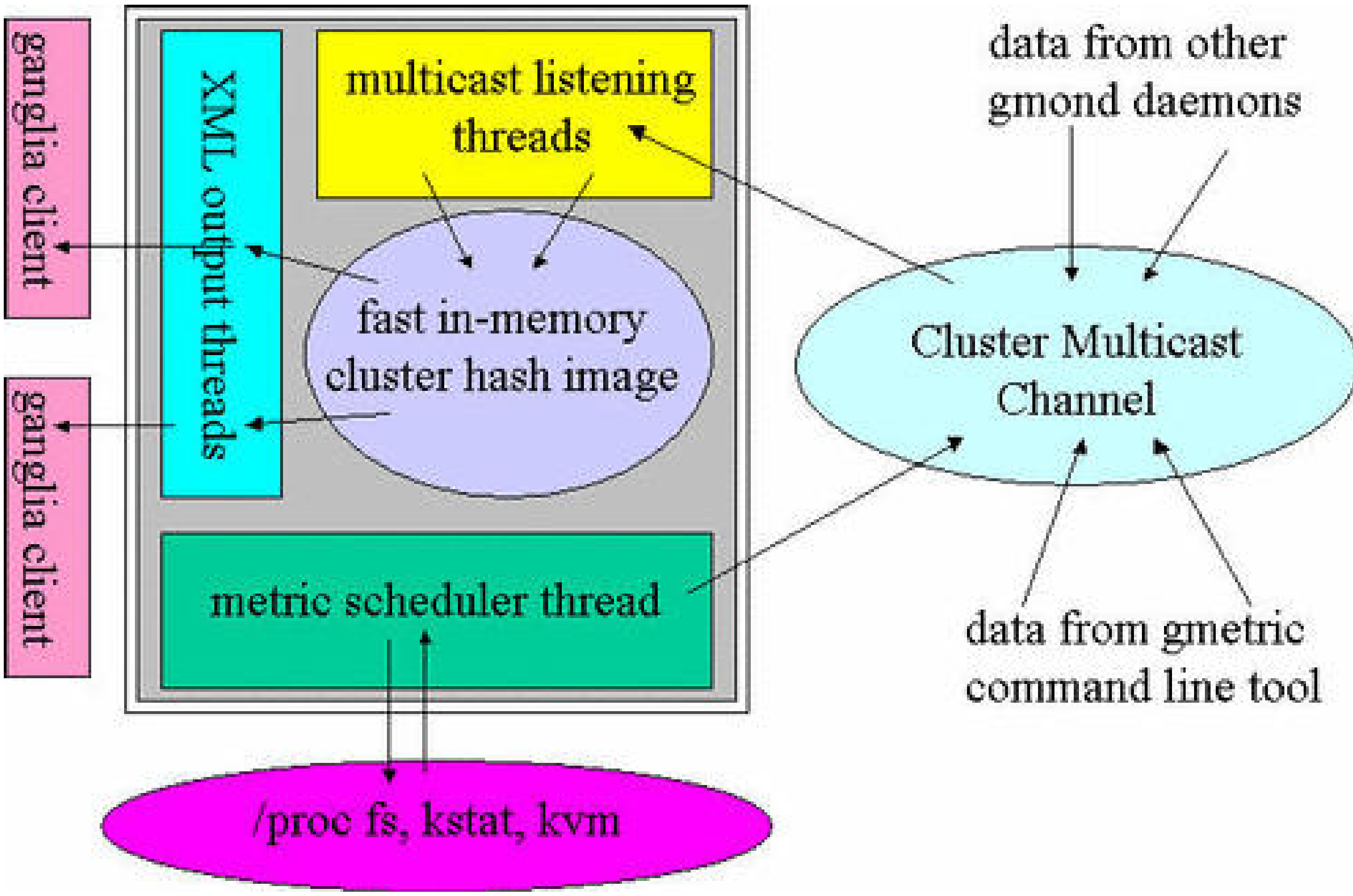
```
telnet remote.cluster.nodename 8649
```

to get a remote XML image (assuming it trusts you!). Also, to help you remember the port number, 8649 is U*N*I*X on a phone keypad.

2.2.1.1.4. Gmond Will Listen to What Gmetric Has to Say Too

You can send custom metrics on the ganglia multicast channel as well. Gmond monitors 24 metrics right out of the box (as of this writing): Number of CPU, CPU Speed, % CPU User, System, Nice Idle, and Absolute Idle, 1,5,15-Minute Load Averages, Total, Free, Shared, Buffered, and Cached Memory, Total and Free Swap Space, Total and Running Processes, OS Name and Release, Hardware Type, and Boottime and System Time. Gmond doesn't assume that these are the only only metrics that you want to monitor. To expand the list of metrics you are monitoring, use the gmetric tool.

2.2.1.2. A Graphical Explanation of Gmond


ganglia client
ganglia client
XML output threads
multicast listening threads
fast in-memory cluster hash image
metric scheduler thread
data from other gmond daemons
Cluster Multicast Channel
data from gmetric command line tool
/proc fs, kstat, kvm

The grey box represents the Ganglia Monitoring Daemon (gmond) with all its components inside: the *metric scheduler thread*, *multicast listening threads*, *fast in-memory hash* and the *XML output threads*.

The *metric scheduler thread* checks the state of the host that gmond is running on and multicasts any relavent changes. Gmond decides what is relavent by using *value and time thresholds*. The metric scheduler remembers what value it last multicast and when it multicast it. If the difference between the last multicast value and the new value is greater than the value threshold, then the metric scheduler will multicast the value. Also, if the time elapsed since the metric value has been multicast is greater than the time threshold it will be sent regardless of its value. The *value and time thresholds* are metric-specific and set based on the metric characteristics.

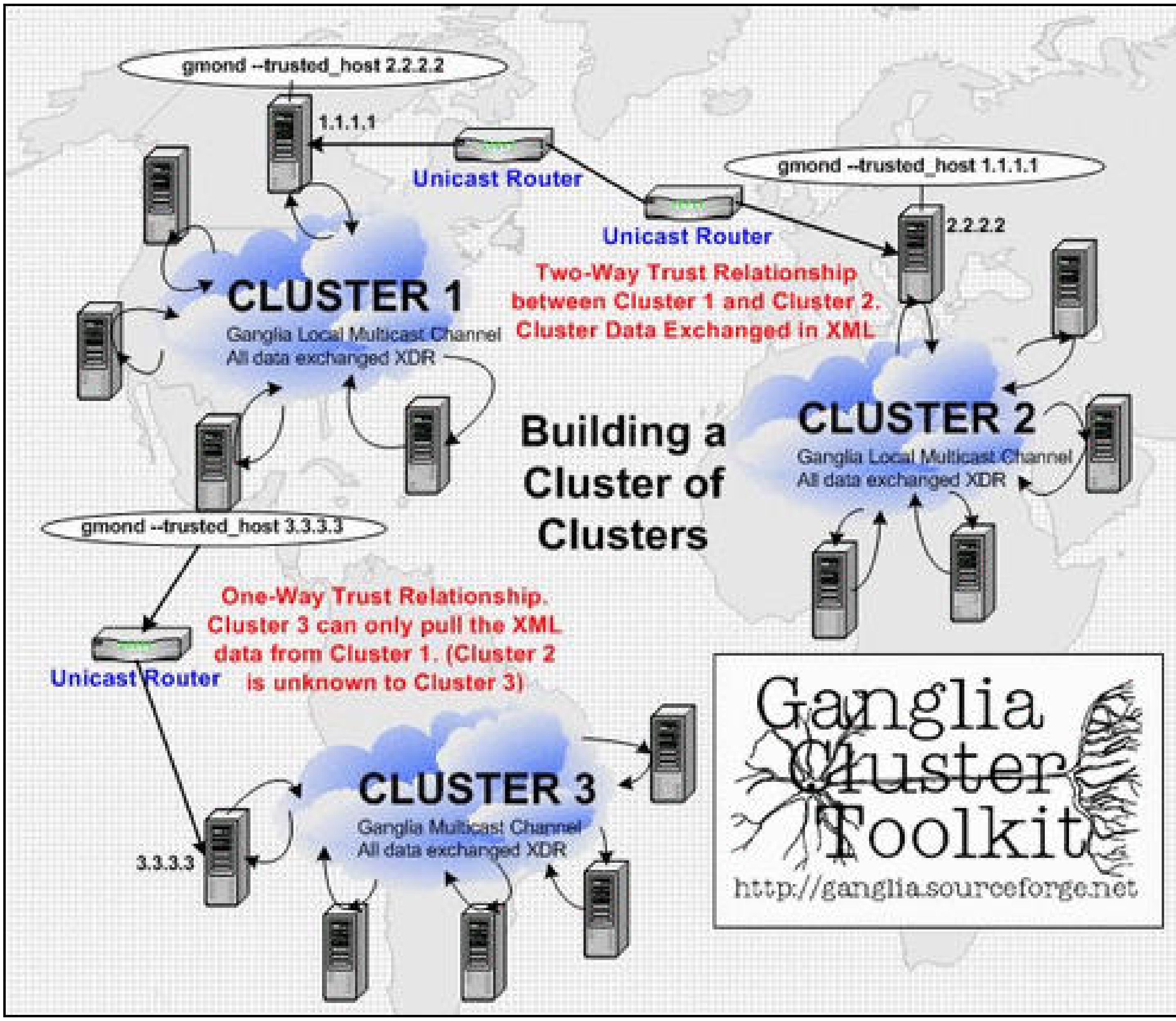
The *multicast listening threads* listen on the ganglia multicast channel for incoming messages including messages from itself (via loopback). All data is stored in the *fast in-memory hash*. This hash holds the data for all hosts sending data on the cluster multicast channel via gmond or gmetric.

The *XML output threads* are responsible for processing incoming connection requests. The XML output threads will only allow hosts in the gmond in-memory hash OR hosts explicitly listed on the gmond commandline to connect. A host connecting to gmond will receive a complete XML description of the state of all hosts multicasting on its local multicast channel. You use telnet if you want to see this description...

```
telnet localhost 8649
```

8649 (U*N*I*X on a phone keypad)is the default port for the XML threads to listen on but it can be changed at compile or run time.

2.2.1.3. Building a *Cluster of Clusters* Using Trust Relationships


gmond --trusted_host 2.2.2.2
1.1.1.1
Unicast Router
gmond --trusted_host 1.1.1.1
Unicast Router
2.2.2.2
Two-Way Trust Relationship between Cluster 1 and Cluster 2. Cluster Data Exchanged in XML
CLUSTER 1
Ganglia Local Multicast Channel
All data exchanged XDR
CLUSTER 2
Ganglia Local Multicast Channel
All data exchanged XDR
Building a Cluster of Clusters
gmond --trusted_host 3.3.3.3
One-Way Trust Relationship. Cluster 3 can only pull the XML data from Cluster 1. (Cluster 2 is unknown to Cluster 3)
Unicast Router
CLUSTER 3
Ganglia Multicast Channel
All data exchanged XDR
3.3.3.3
Ganglia Cluster Toolkit
http://ganglia.sourceforge.net

The vision of the Millennium Project (http://www.millennium.berkeley.edu/) at UC Berkeley (http://www.cs.berkeley.edu/) was to build a campus-wide *"cluster of clusters":* smaller clusters in Civil Engineering, Mathematics, Geology and Geophysics, The Berkeley Multimedia Research Center and over 20 other groups *linked* *together* to a large central cluster in the Computer Science building. This distributed model can also be called *Grid Computing*.

Ganglia easily allows distributed resources to be linked together using *trust relationships* between clusters of computers. The diagram above should be referenced as an example of how to build *trust* between computing resources.

2.2.1.3.1. Trust Relationships Can Be One-Way or Two-Way

Trust relationships are a simple way to connect clusters together in useful ways.

2.2.1.3.1.1. Two-Way Trust Relationships

CLUSTER 1 and *CLUSTER 2* in the diagram above are linked together in a *two-way trust relationship*. All machines on cluster 1 and cluster 2 have the *Ganglia Monitoring Daemon (gmond)* running on them connected to the local multicast channel. This multicast connection allows the nodes in each cluster to update their peers with information about their health, configuration and state. In their default configurations gmonds will only exchange data with other gmonds on their local multicast channel. The *trusted_host* parameter allows gmonds to build trust relationships with hosts outside of the local multicast channel.

To build the two-way trust relationship, the administrator of CLUSTER 1 started the *gmond* process on 1.1.1.1 with the following command line

```
gmond --trusted_host 2.2.2.2
```

The administrator of CLUSTER 2 started the *gmond* process on 2.2.2.2 with the command line

```
gmond --trusted_host 1.1.1.1
```

Now *ganglia* programs and libraries on 2.2.2.2 (Cluster 2) can pull the XML description of Cluster 1 by connecting to the ip address 1.1.1.1 *and* Cluster 1 can pull the XML description from Cluster 2 as well.

2.2.1.3.1.2. One-Way Trust Relationships

In the example diagram above, the administrator of Cluster 1 has also made a node on Cluster 3 a *trusted host*. The administrator of Cluster 3 has not opened his cluster to Cluster 1. While Cluster 3 applications can pull the XML description of Cluster 1 state, Cluster 1 is not allowed to pull any data from Cluster 3. This is a *one-way trust relationship*.

2.2.1.3.2. Trust is Not Additive

It is important to note that just because *Cluster 1* has a trust relationship with both *Cluster 2* and *Cluster 3*. This does not mean that *Cluster 2* and *Cluster 3* have any trust relationship between them. Cluster administrators must explicitly state a trust relationship with a remote cluster before that relationship is realized.

2.2.1.3.3. Only Unicast Routes Between Clusters is Required

While *ganglia* hinges the power of multicast locally for a cluster, all data exchanged between clusters is a unicast TCP XML stream. No special configuration is required of the routers between two clusters.

2.2.1.3.4. Trust Relationship Do Not Allow Others to Exec Jobs on your Cluster

The ganglia execution environment is completely separate from the ganglia monitoring core. The *Ganglia Monitoring Daemon (gmond)* is only responsible for real-time *monitoring* of clusters. The security model for executing jobs will require a completely different set prerequisites (cluster key exchanges, user-authentication, etc).

3. Installing the Ganglia Components

3.1. Ganglia Execution Environment Installation

The ganglia execution environment is comprised of two separate software components: *AUTHD* and *GEXEC*. I recommend that you learn more about the execution components before you begin the installation.

> **Note:** GEXEC relies on AUTHD so AUTHD must be installed on any machine you want to run GEXEC.

> **Note:** The ganglia execution components have only been tested on Linux and are very likely not work on any other OS at this time. We may port it to other OSes when we have a chance but it's not a priority at this time

To simplify the directions, I'm going to use syntax that assumes you have SSH installed. If you don't have SSH installed just interpret *ssh <hostname>* to mean "run a command on <hostname>" and *scp <hostname>* to mean "copy a file to <hostname>"

3.1.1. Installing the authd component

1. Create your cluster-wide RSA public/private key pairs using OpenSSL (http://www.openssl.org/).

   ```
   prompt# openssl genrsa -out auth_priv.pem
   prompt# chmod 600 auth_priv.pem
   prompt# openssl rsa -in auth_priv.pem -pubout -out auth_pub.pem
   ```

   The file *auth_priv.pem* is your *private* cluster key and the file *auth_pub.pem* is your *public* cluster key. These files will be created in the directory that you run the openssl commands in.

2. Install your *public* cluster key on any node which will be *receiving* GEXEC execution requests (most likely every node in your cluster). For example...

   ```
   prompt# scp auth_pub.pem bar1:/etc/auth_pub.pem
   prompt# scp auth_pub.pem bar2:/etc/auth_pub.pem
   prompt# scp auth_pub.pem bar3:/etc/auth_pub.pem
   ```

3. Install your *private* cluster key on any node which will *launch* GEXEC requests. This will likely be your *frontend* node(s). If you want to *launch* GEXEC from your nodes they will need your private cluster key as well as the public cluster key. For example, if you are going to spawn GEXEC jobs from bar1 and bar2 then...

   ```
   prompt# scp auth_priv.pem bar1:/etc/auth_priv.pem
   prompt# scp auth_priv.pem bar2:/etc/auth_priv.pem
   ```

   Warning

   If someone has access to your private cluster key, they will be able to run on any machine in your cluster with the corresponding public key installed. They will also be able to masquerade as any user on the system as well. Double and triple-check that your private key(s) is set mode 0600 and owned by root. If ever you think your cluster private key is compromised redo steps 1-3 above.

4. You need to install authd on every node you want to run GEXEC on. If your cluster nodes have access to the public internet, you only need to run...

   ```
   prompt# ssh bar1 rpm -Uvh http://prdownloads.sourceforge.net/ganglia/authd-0.2.0-1.i386.rpm
   prompt# ssh bar2 rpm -Uvh http://prdownloads.sourceforge.net/ganglia/authd-0.2.0-1.i386.rpm
   prompt# ssh bar3 rpm -Uvh http://prdownloads.sourceforge.net/ganglia/authd-0.2.0-1.i386.rpm
   ...
   ```

   ... on every node in your cluster. Otherwise, you can just download the latest authd RPM (http://prdownloads.sourceforge.net/ganglia/authd-0.2.0-1.i386.rpm) and install it on all your nodes.

Note: If you are going to compile Authd from source, you will to download and install the libe library available from the front page of the ganglia web site (http://ganglia.sourceforge.net/)

Here are the files that will be installed from the authd RPM

```
prompt# rpm -ql authd-0.2.0-1
/etc/init.d/authd
/usr/include/auth.h
/usr/lib/libauth.a
/usr/sbin/authd
```

3.1.2. Installing the Ganglia Execution (GEXEC) component

Note: If you are going to compile Gexec from source, make sure to add the --enable-ganglia flag to the configure script. This ensures that the execution environment connects to the monitoring core instead of working as a stand-alone component. Compiling with the --enable-ganglia flag requires that libganglia be installed on the compile machine. You can download libganglia from http://ganglia.sourceforge.net (http://ganglia.sourceforge.net/).

```
prompt# ./configure --enable-ganglia
loading cache ./config.cache
checking for a BSD compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking whether make sets ${MAKE}... yes
checking for working aclocal... found
checking for working autoconf... found
checking for working automake... found
checking for working autoheader... found
checking for working makeinfo... found
checking for gcc... gcc
checking whether the C compiler (gcc  ) works... yes
checking whether the C compiler (gcc  ) is a cross-compiler... no
checking whether we are using GNU C... yes
checking whether gcc accepts -g... yes
checking for a BSD compatible install... /usr/bin/install -c
checking for ranlib... ranlib
checking for gawk... gawk
checking whether ln -s works... yes
Use of Ganglia for node selection enabled
...
```

Note: Authd must be installed on every node you install GEXEC on.

If your cluster nodes have access to the public Internet, run the following on each node in your cluster...

```
prompt# ssh bar1 rpm -Uvh http://prdownloads.sourceforge.net/ganglia/gexec-0.3.3-1.i386.rpm
prompt# ssh bar2 rpm -Uvh http://prdownloads.sourceforge.net/ganglia/gexec-0.3.3-1.i386.rpm
prompt# ssh bar3 rpm -Uvh http://prdownloads.sourceforge.net/ganglia/gexec-0.3.3-1.i386.rpm
...
```

If your cluster nodes are on a private network, you will need to download the latest RPM for gexec (http://prdownloads.sourceforge.net/ganglia/gexec-0.3.3-1.i386.rpm) and install it on each node.

Here are the files that will be installed from the GEXEC RPM. As you can see from the gexec file placed in the `/etc/xinetd.d/` directory, gexecd is an (x)inetd daemon which by default listens on port 2875/tcp

```
prompt# rpm -ql gexec-0.3.3-1
/etc/xinetd.d/gexec
/usr/bin/gexec
/usr/include/gexec_lib.h
/usr/lib/libgexec.a
/usr/sbin/gexecd
```

3.2. Ganglia Monitoring Core Installation

If you don't understand how each component of the Monitoring Core work together, then I recommend that you read the documentation of the components

3.2.1. Installation from Source

1. Your machine and network must be multicast-enabled to run ganglia. All Linux distributions that I know of keep multicast in the default kernel configuration. Multicast takes up very little space in the kernel. To see if you machine is multicast-enabled...

   ```
   prompt> cat /proc/net/igmp
   Idx     Device    : Count Querier       Group    Users Timer    Reporter
   1       lo        :     0      V2
                                   010000E0     1 0:F1BFBABB               0
   2       eth0      :     1      V2
                                   010000E0     1 0:F1BFBABC               0
   ```

If you don’t see any output, then your kernel is likely not multicast-enabled. By the way, IGMP stands for Internet Group Management Protocol

If your machines are all on the same switch then you are in luck: gmond is ready for use without any commandline tweaks. However, if the machines in your cluster are seperated by a router, then you will need to set the *--mcast_ttl* parameter of gmond to be higher than the default of 1. Set the multi-cast Time-To-Live (TTL) to be one greater than the number of hops (routers) between the hosts. Also, you will also have to make sure that the routers are configured to pass along the multicast traffic.

> **Note:** Some users have reported problems on machines with no "default" route. Gmond reports the error *mcast_join() setsockopt() error: No such device* and then exits at startup. The workaround is to create a static route for the ganglia multicast channel.
>
> ```
> prompt> route add -host 239.2.11.71 dev eth0
> ```
>
> Replace *eth0* above with the name of the network interface you want to send ganglia multicast traffic on

2. Download the source tarball from http://ganglia.sourceforge.net/
3. Unzip the distribution

```
prompt> gunzip < ganglia-monitor-core-2.3.1b4.tar.gz | tar -xvf -
ganglia-monitor-core-2.3.1b4/
ganglia-monitor-core-2.3.1b4/Makefile.in
ganglia-monitor-core-2.3.1b4/README
ganglia-monitor-core-2.3.1b4/stamp-h.in
ganglia-monitor-core-2.3.1b4/AUTHORS
ganglia-monitor-core-2.3.1b4/COPYING
ganglia-monitor-core-2.3.1b4/ChangeLog
ganglia-monitor-core-2.3.1b4/INSTALL
ganglia-monitor-core-2.3.1b4/Makefile.am
ganglia-monitor-core-2.3.1b4/NEWS
ganglia-monitor-core-2.3.1b4/acconfig.h
ganglia-monitor-core-2.3.1b4/acinclude.m4
ganglia-monitor-core-2.3.1b4/aclocal.m4
ganglia-monitor-core-2.3.1b4/config.h.in
...
```

4. Jump into the distribution directory

```
prompt> cd ganglia-monitor-core-2.3.1b4
```

5. Run the configuration script

> **Note:** You must add the *--disable-shared* and *--enable-static* configure flags if you running on AIX

```
prompt> ./configure
creating cache ./config.cache
checking for a BSD compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking whether make sets ${MAKE}... yes
checking for working aclocal... found
checking for working autoconf... found
checking for working automake... found
checking for working autoheader... found
checking for working makeinfo... found
checking host system type... i686-pc-linux-gnu
checking for gcc... gcc
...
creating ./config.status
creating Makefile
creating lib/Makefile
creating gmond/Makefile
creating gmetric/Makefile
creating ganglia.spec
creating config.h
linking ./gmond/machines/linux.c to gmond/machine.c
```

6. Run make

```
prompt> make
make  all-recursive
make[1]: Entering directory '/tmp/mas/ganglia-monitor-core-2.3.1b4'
Making all in lib
make[2]: Entering directory '/tmp/mas/ganglia-monitor-core-2.3.1b4/lib'
/bin/sh ../libtool --mode=compile gcc -DHAVE_CONFIG_H -I. -I. -I..    -I. -O2  -D_REENTRANT -

Wall -Wshadow -Wpointer-arith -Wcast-align -Wstrict-prototypes -D_GNU_SOURCE -c daemon_inetd.
c
mkdir .libs
gcc -DHAVE_CONFIG_H -I. -I. -I.. -I. -O2 -D_REENTRANT -Wall -Wshadow -Wpointer-arith -
Wcast-a
lign -Wstrict-prototypes -D_GNU_SOURCE -c daemon_inetd.c  -fPIC -DPIC -o .libs/daemon_inetd.l
o
```

```
gcc -DHAVE_CONFIG_H -I. -I. -I.. -I. -O2 -D_REENTRANT -Wall -Wshadow -Wpointer-arith -
Wcast-a
lign -Wstrict-prototypes -D_GNU_SOURCE -c daemon_inetd.c -o daemon_inetd.o >/dev/null 2>&1
mv -f .libs/daemon_inetd.lo daemon_inetd.lo
...
```

7. Once everything is built, install it

```
prompt> make install
Making install in lib
make[1]: Entering directory ‘/tmp/mas/ganglia-monitor-core-2.3.1b4/lib’
make[2]: Entering directory ‘/tmp/mas/ganglia-monitor-core-2.3.1b4/lib’
/bin/sh ../config/mkinstalldirs /usr/lib
/bin/sh ../libtool  --mode=install /usr/bin/install -c libganglia.la /usr/lib/libganglia.la
...
```

8. Start up gmond and make sure it is started at reboot

```
prompt> cp ./gmond.init /etc/rc.d/init.d/gmond
prompt> chkconfig --add gmond
prompt> chkconfig --list gmond
gmond           0:off 1:off 2:on 3:on 4:on 5:on 6:off
prompt> /etc/rc.d/init.d/gmond start
Starting GANGLIA gmond:                                    [  OK  ]
```

9. Test your installation

```
prompt> telnet localhost 8649
```

You should see XML that conforms to the ganglia XML spec

3.2.2. Installation Using RPM

1. Your machine and network must be multicast-enabled to run ganglia. All Linux distributions that I know of keep multicast in the default kernel configuration. Multicast takes up very little space in the kernel. To see if you machine is multicast-enabled...

```
prompt> cat /proc/net/igmp
Idx     Device    : Count Querier       Group    Users Timer    Reporter
1       lo        :     0      V2
```

```
                               010000E0     1 0:F1BFBABB              0
2       eth0      :     1      V2
                               010000E0     1 0:F1BFBABC              0
```

If you don’t see any output, then your kernel is likely not multicast-enabled. By the way, IGMP stands for Internet Group Management Protocol

If your machines are all on the same switch then you are in luck: gmond is ready for use without any commandline tweaks. However, if the machines in your cluster are seperated by a router, then you will need to set the --*mcast_ttl* parameter of gmond to be higher than the default of 1. Set the multi-cast Time-To-Live (TTL) to be one greater than the number of hops (routers) between the hosts. Also, you will also have to make sure that the routers are configured to pass along the multicast traffic.

Note: Some users have reported problems on machines with no "default" route. Gmond reports the error *mcast_join() setsockopt() error: No such device* and then exits at startup. The workaround is to create a static route for the ganglia multicast channel.

```
prompt> route add -host 239.2.11.71 dev eth0
```

Replace *eth0* above with the name of the network interface you want to send ganglia multicast traffic on

2. Would you believe me if I told you it’s a simple as installing an RPM on each node in your cluster? Just run this command on each node of your cluster to install the monitoring core. If your cluster nodes have access to the public Internet, just run the following on each node...

```
prompt> rpm -Uvh http://prdownloads.sourceforge.net/ganglia/ganglia-monitor-core-2.3.1b4-
1.i386.rpm
Retrieving http://prdownloads.sourceforge.net/ganglia/ganglia-monitor-core-2.3.1b4-1.i386.rpm
Preparing...                ########################################### [100%]
   1:ganglia-monitor-core   ########################################### [100%]
Starting GANGLIA gmond: [  OK  ]
```

Otherwise you can just download the latest monitoring core RPM (http://prdownloads.sourceforge.net/ganglia/ganglia-monitor-core-2.3.1b4-1.i386.rpm) and install it on each node in your cluster.

3.2.3. Creating a gmond configuration file

While the default options for gmond will work for most clusters, gmond is very flexible and can be customize with the configuration file: /etc/gmond.conf.

/etc/gmond.conf is not required as its absence will only cause gmond to start in a default configuration. Here is a sample of a gmond.conf configuration file...

```
# This is the configuration file for the Ganglia Monitor Daemon (gmond)
# Documentation can be found at http://ganglia.sourceforge.net/docs/
#
# To change a value from it’s default simply uncomment the line
# and alter the value
#####################
#
# The name of the cluster this node is a part of
# default: "unspecified"
# name  "My Cluster"
#
# The multicast channel for gmond to send/receive data on
# default: 239.2.11.71
# mcast_channel 239.2.11.71
#
# The multicast port for gmond to send/receive data on
# default: 8649
# mcast_port    8649
#
# The multicast interface for gmond to send/receive data on
# default: the kernel decides based on routing configuration
# mcast_if  eth1
#
# The multicast Time-To-Live (TTL) for outgoing messages
# default: 1
# mcast_ttl  1
#
# The number of threads listening to multicast traffic
# default: 2
# mcast_threads 2
#
# Which port should gmond listen for XML requests on
# default: 8649
# xml_port     8649
#
# The number of threads answering XML requests
# default: 2
# xml_threads   2
#
# Hosts ASIDE from "127.0.0.1"/localhost and those multicasting
# on the same multicast channel which you will share your XML
# data with.  Multiple hosts are allowed on multiple lines.
# default: none
# trusted_hosts 1.1.1.1 1.1.1.2 1.1.1.3 \
# 2.3.2.3 3.4.3.4 5.6.5.6
#
# The number of nodes in your cluster.  This value is used in the
# creation of the cluster hash.
# default: 1024
# num_nodes  1024
#
# The number of custom metrics this gmond will be storing.  This
# value is used in the creation of the host custom_metrics hash.
# default: 16
# num_custom_metrics 16
#
# Run gmond in "mute" mode.  Gmond will only listen to the multicast
# channel but will not send any data on the channel.
# default: off
# mute on
#
# Run gmond in "deaf" mode.  Gmond will only send data on the multicast
# channel but will not listen/store any data from the channel.
# default: off
# deaf on
#
# Run gmond in "debug" mode.  Gmond will not background.  Debug messages
# are sent to stdout.  Value from 0-100.  The higher the number the more
# detailed debugging information will be sent.
# default: 0
# debug_level 10
#
# If you don’t want gmond to setuid, set this to "on"
# default: off
# no_setuid  on
#
# Which user should gmond run as?
# default: nobody
# setuid     nobody
#
# If you do not want this host to appear in the gexec host list, set
# this value to "on"
# default: off
# no_gexec   on
#
# If you want any host which connects to the gmond XML to receive
# data, then set this value to "on"
# default: off
# all_trusted on
```

If you want to customize the operation of gmond, simply edit this file and save it to /etc/gmond.conf. You can create multiple gmond configurations by writing the configuration file to a different file, say /etc/gmond_test.conf, and the using the --config option of gmond to specify which configuration file to use.

```
# gmond --config /etc/gmond_test.conf
```

would start gmond with the settings in /etc/gmond_test.conf

3.3. Installation of the Ganglia PHP/RRD Web Client

To see a live demo of the Ganglia Client tune your web browser to http://ganglia.mrcluster.org/

3.3.1. Installing prerequisite software

1. You must have a web server installed which correctly parses PHP files. For more information see http://www.php.net (http://www.php.net/)
2. You need to install a ganglia monitoring daemon (gmond) on your web server so that it can listen to the ganglia cluster multicast channel. Visit http://ganglia.sourceforge.net/ if you don't have the ganglia monitoring core tarball or RPM
3. Install rrdtool (http://www.rrdtool.com/download.html). Download the latest "Stable Release". Installation is simple.

   ```
   gunzip < rrdtool-1.0.33.tar.gz | tar -xvf -
   cd rrdtool-1.0.33
   ./configure
   make
   make site-perl-install
   make install
   ```

   (Note: you may need to be root to install rrdtool into your site-perl directory)

4. Install expat (http://expat.sourceforge.net (http://expat.sourceforge.net/)) if you don't already have it installed. Newer Linux installations have it installed out-of-box. To check if you have it installed, run the command "rpm -qa | grep -i expat". If not... Download the latest tarball expat-1.95.2.tar.gz and install.

   ```
   gunzip < expat-1.95.2.tar.gz | tar -xvf -
   cd expat-1.95.2
   ```

```
./configure
make
make install
```

5. Install the perl XML::Parser if you don’t already have it (http://wwwx.netheaven.com/~coopercc/xmlparser/intro.html). Newer Linux installations have it installed out-of-box. To check if you have it installed, run the command "rpm -qa | grep -i parser". If not... Downlaad the latest tarball XML-Parser-2.30.tar.gz and install.

```
gunzip < XML-Parser-2.30.tar.gz | tar -xvf -
cd XML-Parser-2.30
perl Makefile.PL
make
make install
```

3.3.2. Configuration

Note: all files that I refer to from here on are relative to the ganglia-php-rrd-client-x.y.z directory.

1. Check the first few lines of the ganglia-rrd.pl file for configuration options. The default will work for almost everyone. All data is by default written to the /var/log/ganglia directory. If you don’t plan on running the ganglia-rrd.pl daemon as root.. you may need to change the directory OR create that directory as root and make it writable by the ganglia-rrd.pl daemon user. Either way.
2. Check the first few line of the ./web/graph.php file for configuration options. Make sure the the $rrd_dir variable points to the SAME directory as your ganglia-rrd.pl script ("/var/log/ganglia/rrds" by default). Secondly, make sure that the variable $rrdtool points to your "rrdtool" binary. It installs in (/usr/local/rrdtool-x.y.z/bin/rrdtool) by default).

3.3.3. Installation

1. Copy the web files to a directory on your web server.

```
cd ./web
cp * /var/www/html/ganglia
```

2. Make the directory /var/log/ganglia owner root mode 755

```
mkdir /var/log/ganglia
```

```
chown root /var/log/ganglia
chmod 755 /var/log/ganglia
```

3. Copy the ganglia-rrd.pl daemon to /var/log/ganglia

   ```
   cd ..
   cp ganglia-rrd.pl /var/log/ganglia
   ```

   Note: You can put the ganglia-rrd.pl daemon anywhere on your machine you like. The rc file in the next step, Step 4, assumes that it can find it there. If you move the location of ganglia-rrd.pl then make sure you modify the rc file in the next step.

4. Make sure the ganglia-rrd.pl daemon is started at startup.

   ```
   cp ganglia-php-rrd /etc/rc.d/init.d
   chkconfig --add ganglia-php-rrd
   ```

   Note: use the file ganglia_rrd for SuSE Linux

5. Fire up the ganglia-rrd.pl daemon

   ```
   /etc/rc.d/init.d/ganglia-php-rrd start
   ```

   Note: Having multiple ganglia-rrd.pl daemon running at once will make your rrd data incorrect. Only one daemon should be running.

6. Point your web browser to the directory you copied the web files to and check that it's working

3.3.4. Debugging

1. Make sure your web server is up and running. If not, start it and try again. Check your web server logs to see if there is any helping info there.
2. Make sure the ganglia-rrd.pl daemon is running

   ```
   ps -auxw | grep ganglia-rrd.pl
   ```

   If not, then check /var/log/ganglia/ganglia-rrd-errors for clues if any. Also, make sure that the daemon is being started at startup.

3. Check in the directory /var/log/ganglia/rrds to ensure the all the Round-Robin databases are being built correctly. Make sure the permissions are readable by user "nobody" since most web servers run as that user
4. Double-check that you have the configuration options in graph.php and ganglia-rrd.pl correct.
5. Send an email to ganglia-general@lists.sourceforge.net and ask for some help

I hope that you find this tool helpful for monitoring your cluster. Feel free to email me any questions, comments, patches, etc.

4. Using the Ganglia Components

4.1. Using GEXEC

4.1.1. The GEXEC Commandline

-h, --help

Gexec will output help

-n, --nprocs

Used to specify the number of cluster nodes to spawn the command on. If you want to run a job on the entire cluster set n to 0 (zero). This is the only required variable on the commandline

-d, --detached

The gexec client will spawn the remote jobs and then detach. Typically gexec does not exit until all remote processes have exited. This option is very useful to start processes on remote machines which will not exit (e.g. daemons).

-p, --prefix-type

Valid arguments are `(none|ip|vnn|host)`. The default is vnn. This is the string will will be prepended to the gexec output. `None` means no data is prepended. `ip` means the I.P. address of the remote gexec host will be prepended. `vnn` (virtual node number) means that the unique numerical host identifier will be prepended. `host` will prepend the data with the hostname.

-f, --fanout

Gexec uses an n-ary tree for communication with remote processes. The default is a binary tree. This option allows you to change the way the tree is created. A fanout of 4 would be an 4-ary tree, for example.

For example, to run `uptime` on ten nodes in your cluster and have the output prefixed with the hostname run...

```
prompt> gexec -n 10 -p host uptime
mm88.Millennium.Berkeley.EDU   4:40pm  up  4:48,  0 users,  load average: 1.00, 1.00, 1.11
mm98.Millennium.Berkeley.EDU   4:40pm  up  4:47,  0 users,  load average: 0.00, 0.00, 0.07
mm67.Millennium.Berkeley.EDU   4:40pm  up  4:46,  0 users,  load average: 0.00, 0.00, 0.17
mm83.Millennium.Berkeley.EDU   4:40pm  up  4:48,  0 users,  load average: 1.00, 1.00, 1.12
mm38.Millennium.Berkeley.EDU   4:39pm  up 42 days,  5:51,  0 users,  load average: 1.00, 1.00, 1.10
mm87.Millennium.Berkeley.EDU   4:40pm  up  4:48,  0 users,  load average: 0.00, 0.00, 0.08
mm92.Millennium.Berkeley.EDU   4:40pm  up  4:47,  0 users,  load average: 0.00, 0.00, 0.09
mm86.millennium.Berkeley.EDU   4:40pm  up  4:48,  0 users,  load average: 1.00, 1.00, 1.08
mm56.Millennium.Berkeley.EDU   4:40pm  up  4:48,  3 users,  load average: 0.00, 0.00, 0.64
mm42.Millennium.Berkeley.EDU   4:40pm  up  4:46,  0 users,  load average: 0.00, 0.00, 0.24
```

4.1.2. Explicitly Setting Nodes

GEXEC can be used interactively using the gexec client or programmatically using the GEXEC library, libgexec.a. With the client, node selection can be done in one of two ways. It can done by explictly naming a set of nodes using the GEXEC_SVRS environment variable:

```
# export GEXEC_SVRS="tgl0 tgl1 tgl2 tgl3"
# gexec -n 4 hostname
 1 tgl1
 3 tgl3
 0 tgl0
 2 tgl2
```

If you specify less, than the number of GEXEC_SVRS listed then the first n nodes are chosen

```
# export GEXEC_SVRS="tgl0 tgl1 tgl2 tgl3"
# gexec -n 2 hostname
 0 tgl0
 1 tgl1
```

4.1.3. Using a Dynamic Real-Time Node List

Alternatively, node selection can also be done by specifying one or more potential gmond servers to query. The advantage is using this method is that your host list is load-balanced and only includes hosts that are up and have gexec installed.

> **Note:** The real-time, load-balancing sort value is the 1 min. Load minus the number of CPUs for the *number of available CPUs*. The nodes with the most available CPUs are listed at the top. Also, only nodes which are up will be listed preventing any gexec jobs being sent to hosts that are down.

The first gmond server that is both up and returns a non-empty set of nodes will be used to provide the list of nodes. This allows you to list redundant gmond servers in case of network problems or gmond failures.

The syntax for the GEXEC_GMOND_SVRS variable is a space-delimited list of hosts. The host syntax is *host*:*port*. If a port is not specified, then port 8649 (the default gmond XML port) is assumed. For example...

```
# export GEXEC_GMOND_SVRS="tgl1 tgl3:8500"
# gexec -n 0 hostname
 1 tgl1
 4 tgl4
 3 tgl3
 0 tgl0
 2 tgl2
```

...would query the gmond on tgl1 port 8649 first. If that gmond failed to return information then tgl3 would be queried on port 8500

> **Tip:** If you want to monitor a node but do not want it to show up in the list of hosts returned by gmond for gexec use, simply start gmond on that node with the --no_gexec option.

4.2. Using the Ganglia Monitoring Core

4.2.1. The Ganglia Metric Tool (gmetric)

The *Ganglia Metric Tool* (gmetric) allows you to easily monitor any arbitrary host metrics that you like expanding on the core metrics that gmond measures out-of-the-box.

If you want help with the gmetric sytax, simply use the “help” commandline option

```
prompt> gmetric --help
gmetric 2.3.1b4

Purpose:
  The Ganglia Metric Client (gmetric) announces a metric
  value to all Ganglia Monitoring Daemons (gmonds) that are listening
  on the cluster multicast channel.

Usage: gmetric [OPTIONS]...
   -h         --help                  Print help and exit
   -V         --version               Print version and exit
   -nSTRING   --name=STRING           Name of the metric
   -vSTRING   --value=STRING          Value of the metric
   -tSTRING   --type=STRING           Either string|int8|uint8|int16|uint16|int32|uint32|float|doubl
   -uSTRING   --units=STRING          Unit of measure for the value e.g. Kilobytes, Celcius
   -cSTRING   --mcast_channel=STRING  Multicast channel to send/receive on (default='239.2.11.71')
   -pINT      --mcast_port=INT        Multicast port to send/receive on (default=8649)
   -iSTRING   --mcast_if=STRING       Network interface to multicast on e.g. 'eth1' (de-
fault='kernel decides')
   -lINT      --mcast_ttl=INT         Multicast Time-To-Live (TTL) (default=1)
```

The gmetric tool formats a special multicast message and sends it to all gmonds that are listening. Your job is simpler: tell gmetric what data you want sent.

All metrics in ganglia have a *name*, *value*, *type* and optionally *units*. For example, say I wanted to measure the temperature of my CPU (something gmond doesn't do) then I could multicast this metric with *name*="temperature", *value*="63", *type*="int16" and *units*="Celcius".

Assume I have a program called `cputemp` which outputs in text the temperature of the CPU

```
prompt> cputemp
63
```

I could easily send this data to all listening gmonds by running

```
prompt> gmetric --name temperature --value `cputemp` --type int16 \
--units Celcius
```

Check the exit value of gmetric to see if it successfully sent the data: 0 on success and -1 on failure.

To constantly sample this *temperature* metric, you just need too add this command to your cron table.

4.2.2. The Ganglia Cluster Status Tool (gstat)

The Ganglia Cluster Status Tool (gstat) is a commandline utility that allows you to get status report for your cluster. With time, it will be a more flexible way to query a gmond running locally or remotely.

Commandline Options

To get the commandline options simply run...

```
prompt> gstat --help
gstat 2.3.1b4

Purpose:
  The Ganglia Status Client (gstat) connects with a
  Ganglia Monitoring Daemon (gmond) and output a load-balanced list
  of cluster hosts

Usage: gstat [OPTIONS]...
   -h         --help             Print help and exit
   -V         --version          Print version and exit
   -a         --all              List all hosts.  Not just hosts running gexec (default=off)
   -d         --dead             Print only the hosts which are dead (default=off)
   -m         --mpifile          Print a load-balanced mpifile (default=off)
   -1         --single_line      Print host and information all on one line (default=off)
   -l         --list             Print ONLY the host list (default=off)
   -iSTRING   --gmond_ip=STRING  Specify the ip address of the gmond to query (default='127.0.0.1')
   -pINT      --gmond_port=INT   Specify the gmond port to query (default=8649)
```

Running gstat without any parameters will cause it print a load-balanced (least-loaded host first) list of all the hosts running gmond along with the process, load, and CPU information. If you want to see which hosts are down in your cluster, use the `--dead` gstat option. You can also have gstat produce a dynamic load-balanced mpimachine file with the `--mpifile` option.

Gstat Examples

Get a load-balanced list of hosts that are up...

```
prompt> gstat
CLUSTER INFORMATION
       Name: unspecified
      Hosts: 97
Gexec Hosts: 73
 Dead Hosts: 0
  Localtime: Mon Apr 22 16:58:43 2002

CLUSTER HOSTS
```

```
Hostname                    LOAD                       CPU              Gexec
 CPUs (Procs/Total) [     1,     5, 15min] [  User,  Nice, System, Idle]

mm92.millennium.berkeley.edu
    4 (    1/   97) [  1.10,  1.19,  0.99] [   5.9,   0.0,   0.5, 100.0] ON
mm98.Millennium.Berkeley.EDU
    4 (    0/   80) [  1.16,  1.67,  1.25] [   4.1,   0.0,   0.2,  98.5] ON
mm91.Millennium.Berkeley.EDU
    4 (    1/   87) [  1.67,  1.78,  1.69] [  25.0,   0.0,   0.7,  74.9] ON
mm75.millennium.berkeley.edu
    4 (    3/  103) [  1.85,  2.54,  1.83] [  72.6,   0.0,   0.2,  50.3] ON
mm67.millennium.Berkeley.EDU
    4 (    4/  112) [  1.89,  2.08,  1.38] [  81.4,   0.0,   0.1,  38.5] ON
mm87.millennium.berkeley.edu
    4 (    4/  112) [  1.95,  1.67,  1.27] [   3.2,   0.0,   0.4,  96.4] ON
mm83.millennium.Berkeley.EDU
    4 (    1/  120) [  2.00,  2.59,  2.24] [  25.0,   0.0,   0.0,  75.0] ON
mm10.millennium.Berkeley.EDU
    2 (    0/   77) [  0.00,  0.06,  0.07] [   0.2,   0.0,   0.0,  99.9] ON
...
```

To get create a dynamic load-balanced mpifile list

```
prompt> gstat --mpifile
mm56.Millennium.Berkeley.EDU:4
mm44.Millennium.Berkeley.EDU:4
mm31.Millennium.Berkeley.EDU:2
mm43.Millennium.Berkeley.EDU:4
mm15.Millennium.Berkeley.EDU:2
...
```

4.2.3. The Ganglia Monitoring C Library (libganglia)

The ganglia library is in it's early stages right now and not ready for production. I'm commenting the code using Doxygen (http://www.doxygen.org/) so you can take a peak at the latest LibGanglia documentation (http://ganglia.sourceforge.net/docs/libganglia/html/)

5. Frequently Asked Questions

5.1. Execution Environment FAQ

None at this time.

5.2. Monitoring Core FAQ

Q: Can I specify more than one trusted_host on the gmond commandline?

A: Yes.

```
prompt# gmond --trusted_host 1.1.1.1 --trusted_host 1.1.1.2 --trusted_host 1.1.1.3
```

Q: Gmond does not start and I get the error "mcast_join() setsockopt() error: No such device". What does that mean?

A: It means there is no route the kernel could find to the ganglia multicast channel. The fix the problem simply run the following before you start gmond

```
prompt# route add -host 239.2.11.71 dev eth0
```

This example assumes you want the multicast data to be sent via eth0.

Q: I'm running gmond on *x* operating system and some metrics are not being reported

A: All metrics are reported on Linux and FreeBSD. AIX, IRIX, and Solaris support is limited and does not include every metric. You can use the gmetric tool to work around this problem for now but in the future hope to add full support for other operating systems.

6. Ganglia Applications

Since gmond exchanges well-defined XML, it is easy to build applications that plug into the monitoring core. Here are some applications that have been built with ganglia.

6.1. The Ganglia PHP/RRD Web Client

You might want to see the Ganglia PHP/RRD Web Client in action live (http://ganglia.mrcluster.org/) before you install this plugin. It’s a good example of what you can do with the data ganglia collects.

6.2. Greg Bruno’s Ganglia Python Class and Client

Greg Bruno at NPACI Rocks (http://rocks.npaci.edu/) has created a Python class and ganglia client.

7. Getting Help

If you need help, a good starting point is the latest ganglia documentation (http://ganglia.sourceforge.net/docs/). If the manual doesn’t answer your question, then the next place to look is browse the ganglia forums (https://sourceforge.net/forum/?group_id=43021) and mailing list archives (https://sourceforge.net/mail/?group_id=43021). It is very possible that another ganglia user has had the same problem as you. If you still haven’t found an answer, then send mail to the ganglia-general@lists.sourceforge.net or to me, the author, directly at massie@cs.berkeley.edu.

If you may have found a bug then browse the bug list (https://sourceforge.net/tracker/?atid=434892&group_id=43021&func=browse) to see if the problem is already being worked on. If there are no similar bugs reported, then please submit a report of the bug (https://sourceforge.net/tracker/?atid=434892&group_id=43021&func=browse) or even better submit a patch for the problem (https://sourceforge.net/tracker/?atid=434894&group_id=43021&func=browse) `:)`

8. Ganglia Monitor Core XML Specification

All communication between gmond and any ganglia client is done in extensible markup language (XML) which is prefixed with a document type definition (DTD). the DTD gives the client an immediate description of exactly what structure the XML data will have.

Getting the cluster XML description from a node is very simple. You could even use telnet as a ganglia client. for example...

```
telnet localhost 8649
```

on a host running gmond would report the following...

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
```

```
<!DOCTYPE GANGLIA_XML [
   <!ELEMENT GANGLIA_XML (CLUSTER)+>
   <!ATTLIST GANGLIA_XML VERSION CDATA #REQUIRED
                         SOURCE  CDATA #REQUIRED>
   <!ELEMENT CLUSTER (HOST)+>
   <!ATTLIST CLUSTER NAME  CDATA #REQUIRED
                     LOCALTIME CDATA #REQUIRED>
   <!ELEMENT HOST (METRIC)+>
   <!ATTLIST HOST NAME     CDATA #REQUIRED
                  IP       CDATA #REQUIRED
                  REPORTED CDATA #REQUIRED>
   <!ELEMENT METRIC EMPTY>
   <!ATTLIST METRIC NAME   CDATA #REQUIRED
                    VAL    CDATA #REQUIRED
                    TYPE (string|int8|uint8|int16|uint16|
                          int32|uint32|float|double) #REQUIRED
                    UNITS CDATA #IMPLIED
                    SOURCE (gmond|gmetric) #REQUIRED>
]>
<GANGLIA_XML VERSION="2.1.2" SOURCE="gmond">
<CLUSTER NAME="unspecified" LOCALTIME="1014959606">
<HOST NAME="dhcp-50-165.Millennium.Berkeley.EDU" IP="169.229.50.165" REPORTED="1014959603">
<METRIC NAME="mem_free" VAL="126532" TYPE="uint32" UNITS="KBs" SOURCE="gmond"/>
<METRIC NAME="mem_cached" VAL="584896" TYPE="uint32" UNITS="KBs" SOURCE="gmond"/>
<METRIC NAME="proc_total" VAL="103" TYPE="uint32" UNITS="procs" SOURCE="gmond"/>
<METRIC NAME="cpu_num" VAL="1" TYPE="uint16" UNITS="CPUs" SOURCE="gmond"/>
<METRIC NAME="swap_total" VAL="530104" TYPE="uint32" UNITS="KBs" SOURCE="gmond"/>
<METRIC NAME="load_fifteen" VAL="0.02" TYPE="float" UNITS="" SOURCE="gmond"/>
...
...
```

9. ChangeLog

2002-05-01 Matt Massie
   * Gmond now has a configuration file for its configuration instead of
     passing commandline arguments

2002-04-26 Asaph Zemach
   * Completely rewrote ./gmond/machines/linux.c to be much more efficient by
     removing the need for 2 threads and pthread mutexes.

```
2002-04-26 Matt Massie
  * Modified net.c setsockopt() functions to be more portable

2002-04-19 Matt Massie
  * Fixed a bug in the .spec file where upgrades failed if gmond was not up
  * Updated autoconf to correctly configure ia64 machines

2002-04-18 Matt Massie
  * Enabled setsockopt SO_REUSEADDR for the mcast_join socket in order allow
    multiple instances of gmond to listen to the same multicast channel

2002-04-17 Matt Massie
  * Added the --list and --single_line option to gstat for output flexibility

2002-04-17 Alan Hagge
  * Added preliminary support for IRIX

2002-04-16 Matt Massie
  * Add the "gexec" metric in preparation for the release of gexec
  * Updated the gexec_cluster() function in libganglia to pull in more data
    from the XML and handle unresolved names and names without domains correctly
  * Updated gstat to print the updated information from gexec_cluster()
  * Added a --no_gexec flag to gmond for hosts that are not part of the
    computation cluster (file servers, frontends, etc)
  * Added a --all_trusted flag where gmond sees ALL hosts as trusted as
    suggested by Martin Knoblauch
  * Removed goto statement in pre_process_node() function to avoid rare looping bug
    thanks to feedback from Mike Snitzer

2002-04-12 Chris Elmquist
  * Updated gmetric.c to check if the name of the metric is empty to prevent
    the problem of blank metrics showing up in gmonds

2002-04-08 Matt Massie
  * Fixed a bug in the barrier code used in gmond
  * Fixed mcast_connect() on non-Linux systems related to the IP_MULTICAST_TTL option
  * Added the "--no_setuid" and "--setuid" flags to provide more euid flexibility

2002-04-05 Preston Smith
  * Added support for AIX machines
```

```
2002-04-04 Dave Wallace
  * Fixed a bug in gmond which caused big-endian architectures to incorrectly
    store multicast data and therefore misreport on the XML port.
  * Added a key value check of the XDR multicast data in ./gmond/listen.c

2002-04-04 Matt Massie
  * Updated ./lib/gexec_func.c gexec_cluster() function to handle hosts with
    no domainname correctly (as pointed out by David Wallace
  * Updated the program commandline options to obey the GNU Coding Standard
  * Added a --debug_level parameter to gmond to allow debugging the daemon
    without recompiling the daemon
  * Updated the --trusted_host option to allow multiple instances of the option
  * Added a --mcast_ttl gmond option to allow you to modify the
    Time-To-Live (TTL) of the outgoing multicast messages

2002-04-02 Neil Spring
  * Updated the gmond Makefile to cleanup the machine.c link on "make clean"

2002-04-02 Doc Schneider
  * Update the way the number of CPUs are collected in order to workaround a
    bug on AMD-based systems.

2002-03-28 Matt Massie
  * Change distribution tree to fit the ganglia model.  All gmond clients are
    in the ./gmond directory and all gregisterd clients in ./gregisterd

2002-03-26 Matt Massie
  * Removed the use of streams (via fdopen) on the XML socket and replaced
    it with write()s on the socket descriptor to work around Linux bug under
    high-stress conditions

2002-03-25 Matt Massie
  * Added thread barriers to gmond initialization to ensure listening threads
    exist before the threads which multicast are created

2002-03-24 Matt Massie
  * Use XML_ParseBuffer() in gexec_cluster() in order to avoid
    double copying of buffers (XML input from gmond). Faster.
```

```
2002-03-22 Matt Massie
  * Updated gexec_cluster_free() to ensure no memory leaks even when
    cluster.num_nodes and cluster.num_dead_nodes equals zero
  * Used setvbuf to ensure the gmond and libganglia are using line buffering

2002-03-21 Matt Massie
  * Updated net.h to include netinet/in.h
  * Changed sockaddr_in_new function in net.c to plug a potential memory leak
  * Added XML_ParserFree() call to gexec_cluster() lib call to plug up
    memory leak
  * Modified ./gmond/server.c to prevent crashes under heavy stress conditions
  * Added fclose() calls to gexec_cluster() to plug a memory leak
  * Added gexec_cluster_free() call to gstat for good measure

2002-03-15 Preston Smith
  * Patched a wrong sysctl to get free memory for freebsd machines

2002-03-15 Matt Massie
  * Added mute and deaf mode for gmond
  * Created a new ganglia-monitor-core-lib distribution for the
    libganglia library.
  * Moved all the documentation to DocBook, added much more information
    and updated/removed what was there.  Output docs to ./docs directory
    of the distribution in both HTML and PDF form.  Also installed
    Doxygen to document libganglia.
  * Add the Ganglia Status Tool (gstat) which allows you to check the status
    of your cluster from the commandline.  Hosts are sorted with least-loaded
    nodes at the top of the list.

2002-03-12 Matt Massie
  * Removed the need for the POSIX mutex in pre_process_node() allowing for
    faster processing of incoming multicast data

2002-03-11 Matt Massie
  * Changed gmond to not count itself as a running process when reporting
    the number of running processes.

2002-03-06 Doc Schneider
  * Changed the way ganglia builds RPMs to support non-root builds as
    suggested by an anonymous SourceForge user
```

```
2002-02-28 Matt Massie
   * Added a new CLUSTER element to the XML with two attributes:
     NAME and LOCALTIME.  Necessary for monitoring clusters in
     many different timezones.

2002-02-27 Matt Massie
   * Fixed the getopt_long() call in gmond.c thanks to feedback from
     Meik Hellmund.  The getopt_long() parameters didn't match the
     switch() statement breaking the "trusted_host" option.
   * Fixed a bug in gmond where connections from untrusted hosts caused
     segfaults.  Error caused by passing datum_free() a NULL pointer in
     server_thread() of ./gmond/server.c.
   * Changed the way transient nameservice errors are handled by
     pre_process_node() in ./gmond/listen.c.  Previously, transient errors
     were retried but now they are treated as errors (although gmond
     will continue trying to resolve the host when it gets a new
     multicast packet from it)
   * Updated the ganglia.spec file to merge gmond and gmetric into a single
     RPM, fixed some small bugs, and updated the RPM information.
   * Changed the gmetric options to also support long options and updated
     the help output (from -h, --help) to be much more descriptive

2002-02-27 Preston Smith
   * Updated the FreeBSD monitoring code to include all metrics
     which are monitored under Linux except number of running processes,
     absolute cpu idle time, and shared memory.  SMP users may find that
     freebsd's cp_time sysctls is not completely accurate under FreeBSD stable
     meaning CPU%s might be inaccurate.  However, it works under FreeBSD-CURRENT.

2002-02-22 Matt Massie
   * Added the getopt source to the ganglia library for system that don't
     have the getopt API available (Solaris, FreeBSD, etc)

2002-02-21 Matt Massie
   * Changed the "safe_host" option to "trusted_host" to make it clearer
   * Added the "num_nodes" and "num_custom_metrics" commandline options

2002-02-20 Matt Massie
   * Completely rewrote the underlying hash library because the original
```

```
    hash functions where over-engineered and had some memory leaks.
    New hash functions are superlight and fast.  Built test program and
    profiled/traced all memory functions using mpatrol.  No leaks.
  * Updated code to catch when transient nameservice errors occur and retry.
    Correctly handle hosts the don't resolve instead of treating as an error.
  * Added a patch submitted by Joshua J England for gmond to correctly report
    the number of CPUs and their speed on alpha architectures
  * Added a patch submitted by Eirikur Hallgrimsson and written by
    Yaroslav Klyukin for gmetric which allows users to chose which network
    interface gmetric multicasts metric data

2002-02-12 Matt Massie
  * Reduced the number of total threads by one by removing the
    for(;;)pause() spin and having the main thread do server work

2002-02-07 Matt Massie
  * created the function my_inet_ntop() function in libganglia to deal with
    the limitations of inet_ntoa in a multi-threaded environment
  * changed the self-organzing behavior of gmond to recognize when a transient
    error occured on a remote gmond process
  * added verbose error checking of gethostbyaddr() in listen.c

2002-02-04 Matt Massie
  * fixed a bug in the cpu_num_func() which reported AMD systems as having
    twice as many CPUs as they really did

2002-02-01 Matt Massie

  * increased the speed of the host security check for the XML port
  * added commandline options for almost all compile-time opts for gmond
  * added a --safe_host option to allow a host outside of the multicast
    channel to connect
  * now gmond strips all quotes (") from gmetric data to keep XML well-formed
  * improved the self-organizing behavior of the gmonds

2002-01-18 Matt Massie

  * changed the name of the distribution from ganglia to
    ganglia-monitor-core to be more descriptive
  * modified ./gmond/server.c in order to keep gmond from crashing
```

```
   when a client closes the connection prematurely
 * added patch to ./gmond/gmond.c from Neil Spring and Brian Youngstrom to
   add command line option and properly setuid to nobody
 * added gmond.init.SuSE to the distribution.  Contributed by Oliver Mossinger.
```

10. License

Ganglia is released under the following BSD license.

11. Special Thanks from Matt

I could never have written ganglia without the input of so many people. I want to make sure that everyone gets credit for the ideas they've shared.

First and foremost, *ganglia* is a tool I created while working as a staff researcher on the UC Berkeley Millennium Project (http://www.millennium.berkeley.edu/). The Millennium Project aimed to develop and deploy a campus-wide *cluster of clusters* to support advanced applications in scientific computing, simulation and modelling. Millennium grew out it's predecessor the NOW Project (http://now.cs.berkeley.edu/). Without the vision of Professor *Dave Culler* (http://www.cs.berkeley.edu/~culler/) the Millennium Project would not have existed. Without the Millennium

Project, Ganglia would not have existed. The many faculty, graduate students and staff working on the project were a source of ideas and feedback for me as ganglia was developing.

Albert Goto, *Eric Fraser* and *Peter Sakosky* worked with me as members of the Millennium staff. All four of us share the same office and on many occasions I've bounced ideas off of them and received great feedback.

Brent Chun (http://www.cs.berkeley.edu/~bnc/) and Matt Welsh (http://www.cs.berkeley.edu/~mdw/) were two graduate students would provided a wealth of good advice about the network programming aspects of ganglia. Brent Chun opened my eyes to the power of multicast in a cluster environment.

Millennium has also worked closely with the NPACI Rocks (http://rocks.npaci.edu/) group at the San Diego Supercomputer Center (http://www.sdsc.edu/). *Greg Bruno, Mason Katz, and Phil Papadopoulos* showed me how to bundle of ganglia into an RPM for distribution. Mason Katz also provided some patches and Greg Bruno wrote the Python class for ganglia.

The Ganglia ChangeLog is littered with names of people who have contributed directly to the development of ganglia. The constant feedback of ganglia users/developers is very much appreciated! Thanks guys!

Lastly, two books I could not have been without was Richard Stevens' (http://www.kohala.com/start/) "Unix Network Programming Network APIs: Sockets and XTI" and Volume 2 "Interprocess Communications". Even though I'll never be able to meet him, I admire him for leaving behind so much quality.

12. History of Ganglia Monitoring Core

I came to Berkeley in October of 1999 to work on the Millennium Project in the UCBerkeley CS division (http://www.millennium.berkeley.edu/). This project aimed to build a campus-wide "cluster of clusters" across 20+ departments and linked to a huge central cluster in the Computer Science department. From the beginning it was clear that managing computing resources distributed all over campus was going to be a challenge...

My very first naive attempt to monitor the clusters used Perl DBI to write state to a MySQL database. It even had a strange name, *shepherd*, because I felt like monitoring hundreds of machines was similar to herding cats. I wasn't taking into consideration the lessons learned from the NOW Project (http://now.cs.berkeley.edu/) that building centralized cluster management tools was not scalable at all. We needed a way to monitor clusters that was as distributed and easy to manage as possible.

In the Fall of 2000, I wrote a prototype monitoring system in Perl named *ganglia* because I wanted to create a sytem that would be our cluster's central nervous system. A daemon called a "dendrite" running on each cluster node parsed the /proc directory and multicast what it found. Separate daemons (I called "axons") listened to the multicast traffic and stored it in an in-memory hash to be queried by a commandline tool. Since multicast data was redundant, we could install as many "axons" as we liked in case one (or more) crashed. I demo'd the system at SC2000 (http://www.sc2000.org/) by building a web interface which queried an "axon" and then displayed the state of our cluster at the NPACI (http://www.npaci.edu/) booth.

With very little work I moved the Perl prototype to C (I love perl) and version 1 of ganglia was born. Version 1 had so many shortcomings though. The API for querying the “axons” was *VERY* kludgey. Programmers who wanted to query the axons directly from their programs where frustrated. The data also did not consider endianess so it was not portable. You could not easily extend the system metrics you were monitoring as everything was hard-coded. The daemons were not multi-threaded but rather relied too much on poll(). I could go on for days about how bad it was but I learned *many* valuable lessons

With time, we were running both “dendrites” AND “axons” on all machines for total redundancy. In the spring of 2001, I decided to build a single daemon which would replace the two daemons (dendrite & axon) running on all the machines. Gmond was born. Ganglia version 2 erased the shortcomings of version 1. It replaced the kludgey API by exporting all information in an open, well-defined XML format. All data exchanged between gmonds is in XDR to remove all endianess problems and make the code very portable. A tool called gmetric which allows administrators to arbitrarily expand (using simple shell scripts) the system metrics they monitored.