**Ganglia**

**Distributed monitoring system**

-Tirumal

---

**Cluster?**

A cluster is a collection of computers which work together in accomplishing a task.

Cluster computing has become a practical choice for high-performance computing (HPC) deployment.
Ganglia?

Ganglia - A real-time cluster monitoring tool that collects information from each computer in the cluster and provides an interactive way to view the performance of the computers and cluster as a whole.

Ganglia like other monitoring tools only provide a way to view but not control the performance of each computer.

Ganglia Architecture
Ganglia - A monitoring tool

Ganglia consists of two parts:
- **gmond** (ganglia monitor daemon)
- **gmetad**

**Gmond:** Runs on every node of the cluster and collects data about the node like CPU load, free memory, disk usage, network traffic, etc.

**Gmetad:** Runs on a head node, gathers the data from all the nodes, and displays it.

Ganglia is scalable as we can gather other metrics of interest, send them to the host and display them.

- It is currently in use on over 500 clusters around the world, can handle clusters with 2000 nodes.
A snapshot of our enhanced Ganglia

- Modifying the source code
- Using the gmetric tool (provided by Ganglia)
Modifying the source code

The Ganglia source code includes three files specific to metrics.

/gmond/key_metrics.h
/gmond/metric.h
/gmond/machines/linux.c

Key_metrics.h

enum {
    cpu_num,
    cpu_speed,
    mem_total,
    swap_total,
    cpu_temp,
    sys_clock,
    mem_free,
    mem_shared,
    mem_buffers,
    cpu_idle,
    swap_free,
    load_one,
    load_five,
    load_fifteen,
    proc_run,
    proc_total,
    ....
}
Metric.h

extern g_val_t cpu_num_func(void);
extern g_val_t cpu_speed_func(void);
extern g_val_t mem_total_func(void);
extern g_val_t swap_total_func(void);
extern g_val_t sys_clock_func(void);
extern g_val_t cpu_idle_func(void);
extern g_val_t load_one_func(void);
extern g_val_t load_five_func(void);
extern g_val_t load_fifteen_func(void);
extern g_val_t proc_run_func(void);
extern g_val_t proc_total_func(void);
extern g_val_t cpu_temp_func(void);

/machines/linux.c

g_val_t cpu_num_func ( void )
{
  static int cpu_num = 0;
  g_val_t val; /* Only need to do this once */ if (! cpu_num) {
    cpu_num = get_nprocs();
  }
  val.uint16 = cpu_num;
  return val;
}

g_val_t cpu_temp_func(void)
{
  val.uint16=34;
  return val;
}
Using gmetric tool

- **Gmetric tool provides an easy way to add metrics.**
- **It is provided with Ganglia.**
- **The metrics added by this tool do not remain after a restart.**

**Syntax:**

```
gmetric --name=<metric name> --value=<valueofmetric> --type=<typeofval> ...
```

**Example:**

```
gmetric --name=cpu_temp --value=30 --type=uint8
```

**Courtesy:** [http://monitor.millennium.berkeley.edu/](http://monitor.millennium.berkeley.edu/)
**RRDtool**

**RRDtool (Round Robin Database tool) is a system to store and display time-series data**

**Creating RRD database**

```
rrdtool create target.rrd --start 1023654125 -s 300
DS:mem:GAUGE:600:0:671744
```

- Tutorial can be found at [http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/tutorial/](http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/tutorial/)