



December 3-6, 2007, Santa Clara Marriott, Santa Clara, CA

Monitoring Your Data Center

Using CIM Statistics Model and Ganglia

Brad Nicholes

Sr. Software Engineer, Novell

bnicholes@novell.com





Agenda Ganglia Monitoring

- Introduction and Overview
- Ganglia Architecture
 - Gmond (Gathering)
 - Gmetad (Aggregating)
 - Round-Robin Database
- Exposing Ganglia Statistics in CIM
 - Gathering Service
 - Aggregation Service
 - Reporting Service
- Module Development & Deployment



Ganglia Introduction and Overview

- Scalable Distributed Monitoring System
- Targeted at monitoring clusters and grids
- Multicast-based Listen/Announce protocol
- Depends on open standards
 - XML
 - XDR compact portable data transport
 - RRDTool - Round Robin Database
 - APR – Apache Portable Runtime
 - Apache HTTPD Server
 - PHP based web interface
- <http://ganglia.sourceforge.net> or <http://www.ganglia.info>



Ganglia Architecture

- Gmond – Statistics gathering agent installed on individual servers
- Gmetad – Statistics aggregation agent installed on one or more specific task oriented servers
- Apache Web Frontend – Statistics presentation and analysis server
- Attributes
 - Multicast – All Gmond nodes are capable of listening to and reporting on the status of the entire cluster
 - Failover – Gmetad has the ability to switch which cluster node it polls for statistical data
 - Lightweight and low overhead statistics gathering and transport
- Ported to multiple platforms (Linux, FreeBSD, Solaris, others)



Deploying Ganglia Monitoring

- See <http://ganglia.sourceforge.net/docs/ganglia.html>
- Install Gmond on all monitored nodes
 - Edit the configuration file
 - Add cluster and host information
 - Configure network `udp_send_channel`, `udp_recv_channel`, `tcp_accept_channel`
 - Start gmond
- Installing Gmetad on an aggregation node
 - Edit the configuration file
 - Add data and failover sources
 - Add grid name
 - Start gmetad
- Installing the web frontend
 - Install Apache httpd server with `mod_php`
 - Copy Ganglia web pages and PHP code to appropriate location
 - Add appropriate authentication configuration for access control

Demo – Ganglia Monitoring System



Gmond – Statistics Gathering Agent

- Built-in metrics
 - Various CPU, Network I/O, Disk I/O and Memory
- Extensible
 - Gmetric – Out-of-process utility capable of invoking command line based metric gathering scripts
 - Loadable modules capable of gathering multiple metrics or using advanced metric gathering APIs
- Built on the Apache Portable Runtime
 - Supports Linux, FreeBSD, Solaris and more...



Gmond – Statistics Gathering Agent

- Automatic discovery of nodes
 - Adding a node does not require configuration file changes
 - Each node is configured independently
 - Each node has the ability to listen to and/or talk on the multicast channel
 - Can be configured for unicast connections if desired
 - Heartbeat metric determines the up/down status
- Interfaces
 - Collection functions – Capable of running specialized functions for gathering statistical data
 - Multicast I/O – Listen/Send for statistical data from/to other nodes in the same cluster
 - Data export listeners – Listen for client requests for cluster statistical data



Gmond – Configuration Example

```
globals {
  daemonize = yes
  setuid = yes
  user = nobody
  debug_level = 0
  max_udp_msg_len = 1472
  mute = no
  deaf = no
  host_dmax = 0 /*secs */
  cleanup_threshold = 300 /*secs */
  gexec = no
}
cluster {
  name = "My Cluster"
  owner = "Administrator"
  latlong = "N37.37 W122.23"
  url = "http://www.moreinfo.org"
}
```

```
udp_send_channel {
  mcast_join = 239.2.11.71
  port = 8649
  ttl = 1
}
udp_rcv_channel {
  mcast_join = 239.2.11.71
  port = 8649
  bind = 239.2.11.71
}
tcp_accept_channel {
  port = 8649
}
```



Gmond – Statistics Collection Groups

- Specify as many collection groups as you like
- Each collection group must contain at least one “Metric” section
- List available metrics by invoking “gmond -m”
- Collection_group section:
 - collect_once – Specifies that the group of static metrics
 - collect_every – Collection interval (only valid for non-static)
 - time_threshold – Max data send interval
- Metric section:
 - Name – Metric name (see “gmond -m”)
 - Value_threshold – Metric variance threshold (send if exceeded)



Gmond – Configuration Example

```
collection_group {
  collect_once = yes
  time_threshold = 20
  metric {
    name = "heartbeat"
  }
}
collection_group {
  collect_once = yes
  time_threshold = 1200
  metric {
    name = "cpu_num"
  }
  metric {
    name = "cpu_speed"
  }
  metric {
    name = "mem_total"
  }
  metric {
    name = "swap_total"
  }
  ...
}
```

```
collection_group {
  collect_every = 20
  time_threshold = 90
  metric {
    name = "load_one"
    value_threshold = "1.0"
  }
  metric {
    name = "load_five"
    value_threshold = "1.0"
  }
  ...
}
collection_group {
  collect_every = 80
  time_threshold = 950
  metric {
    name = "proc_run"
    value_threshold = "1.0"
  }
  metric {
    name = "proc_total"
    value_threshold = "1.0"
  }
}
```



Gmetad – Statistics Aggregation Agent

- Polls a designated cluster node for entire cluster status
 - Data collection thread per cluster
 - Ability to poll gmond or another gmetad for statistical data
- Failover capability
- RRDTool – Storage and trend graphing tool
 - Defines fixed size databases that hold data of various granularity
 - Capable of rendering trending graphs from the smallest granularity to the largest (eg. Last hour vs last year)
 - Never grows larger than the predetermined fixed size
 - Database granularity is configurable through `gmetad.conf`



Gmetad – Configuration Example

```
data_source "my cluster" 10 localhost my.machine.edu:8649
1.2.3.5:8655
data_source "my grid" 50 1.3.4.7:8655 grid.org:8651 grid-
backup.org:8651
data_source "another source" 1.3.4.7:8655 1.3.4.8

trusted_hosts 127.0.0.1 169.229.50.165 my.gmetad.org
xml_port 8651
interactive_port 8652

rrd_rootdir "/var/lib/ganglia/rrds"
```

Round-Robin Database Storage



Round-Robin Database (RRD)

- High performance data logging and graphing system for time series data
- Automatic data consolidation over time
 - Define various Round-Robin Archives (RRA) which hold data points at decreasing levels of granularity
 - Multiple data points from a more granular RRA are automatically consolidated and added to a courser RRA
- Constant and predictable data storage size
 - Old data is eliminated as new data is added to the RRD file
 - Amount of storage required is defined at the time the RRD file is created
- RRDTOol Web Site: <http://oss.oetiker.ch/rrdtool/>



Ganglia Default RRD Definition

- Definition of the Round-Robin Database format is determined at database creation time
- Default Ganglia RRA definitions:
 - RRA #1 – 15 second average for 61 minutes
 - RRA #2 – 6 minute average for 24.4 hours
 - RRA #3 – 42 minute average for 7.1 days
 - RRA #4 – 2.8 hour average for 28.5 days
 - RRA #5 – 24 hour average for 374 days
- Default largest retrievable time series, ~1 year
- Configurable to whatever you want

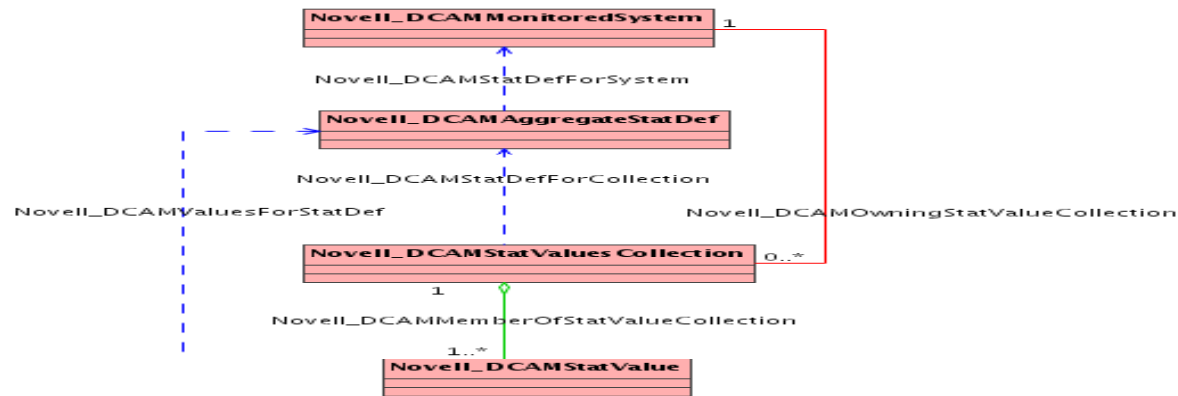


Retrieving Data, Generating Graphs and Interacting with an RRD File

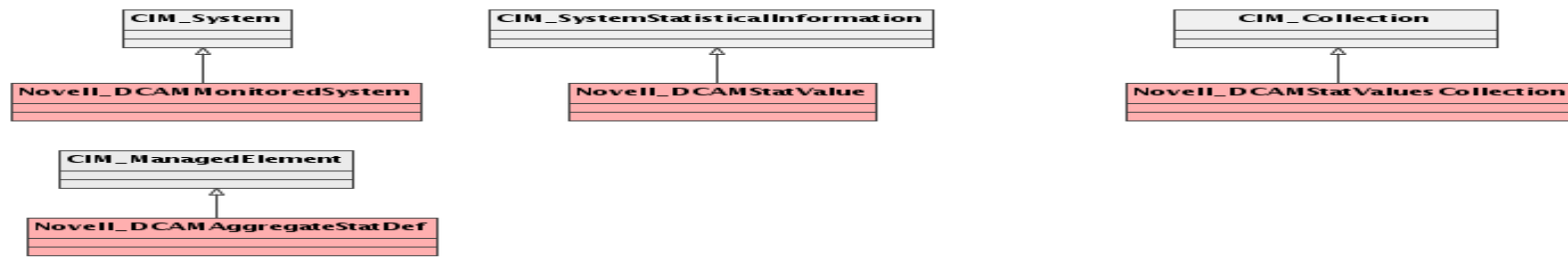
- RRDFetch – Retrieve time series data from an RRD file for a specific time period
- RRDInfo – Print header data from an RRD file in a parsing friendly format
- RRDGraph – Creates a graphical representation of the specified time series data
- RRDUpdate – Feed new data values into an RRD file
- Other APIs – RRDCreate, RRDDump, RRDFirst, RRDLast, RRDLastupdate, RRDResize, ...

Exposing Ganglia Statistics in CIM

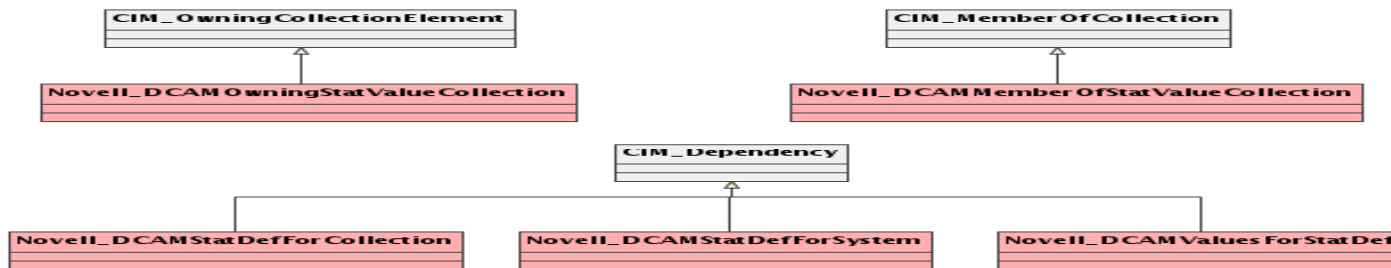
Statistics Reporting Service



Class Inheritance Hierarchy



Association Inheritance Hierarchy

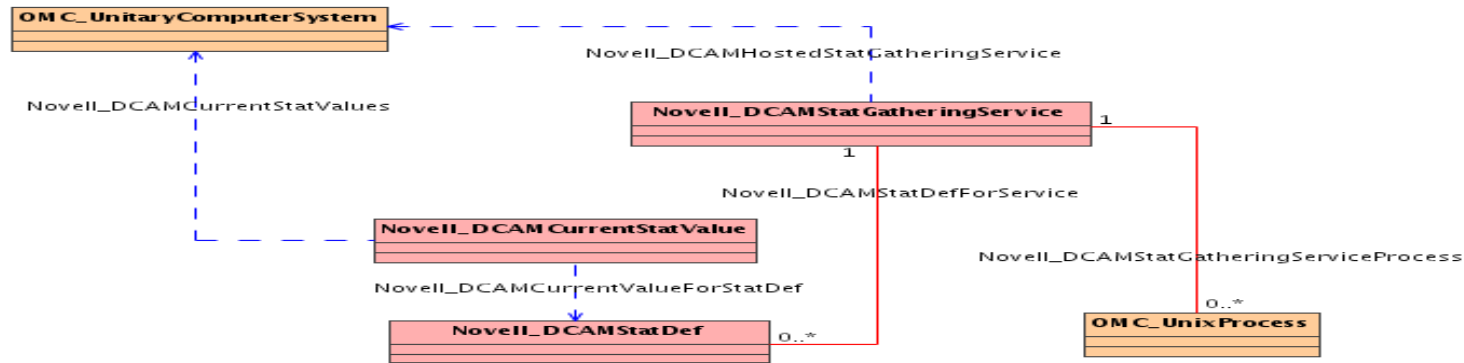




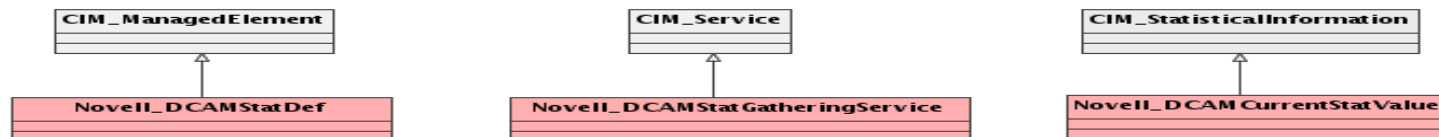
Statistics Reporting Classes

- **Novell_DCAMAggregateStatDef**
 - Represents a statistics definition as viewed by the aggregation service. This includes elements such as Name, Data Type, etc.
 - Definition of a statistic gathered from a monitored computer system
- **Novell_DCAMStatValuesCollection**
 - Represents a set of statistical values for a given time frame associated with a monitored computer system
 - Time series data extracted from an RRD file
- **Novell_DCAMStatValue**
 - Represents a single statistical value associated with a values collection for a monitored computer system
 - Single statistical value extracted from an RRD file

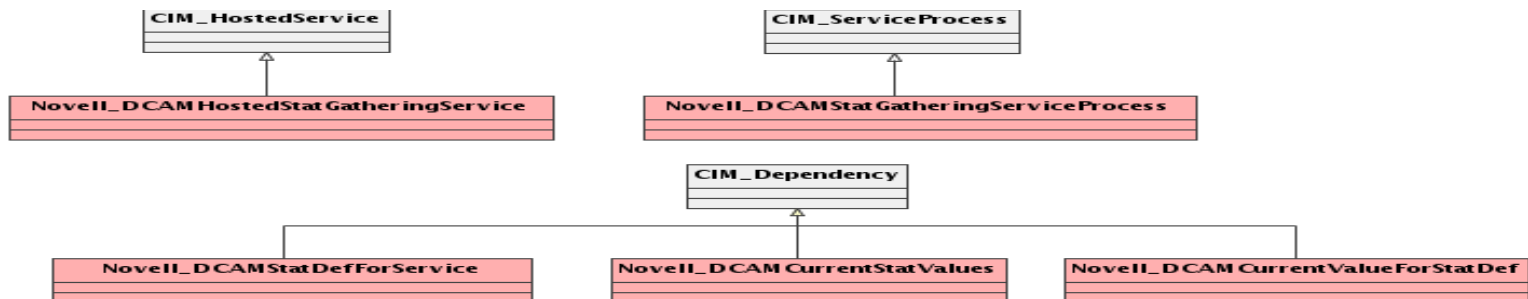
Gmond – Statistics Gathering Service



Class Inheritance Hierarchy



Association Inheritance Hierarchy



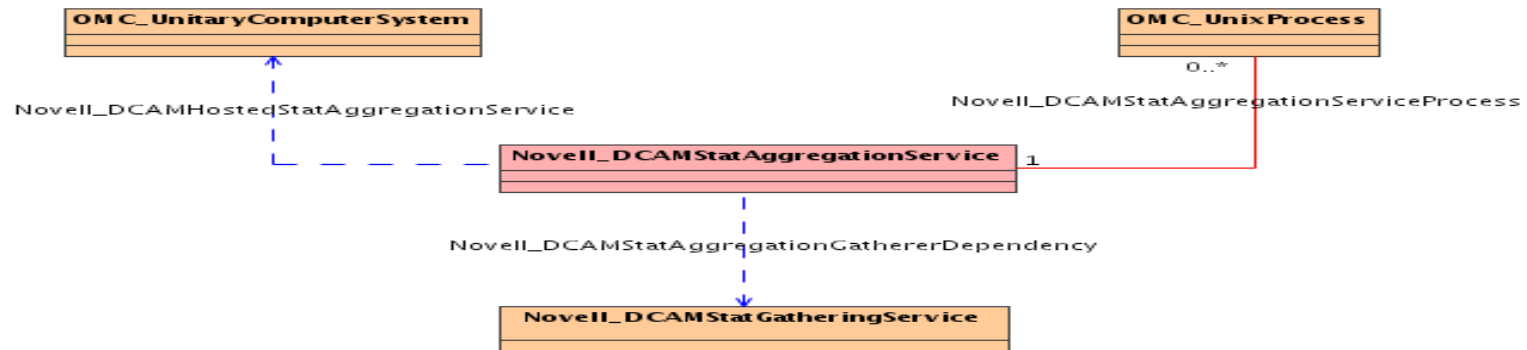


Gmond – Statistics Gathering Classes

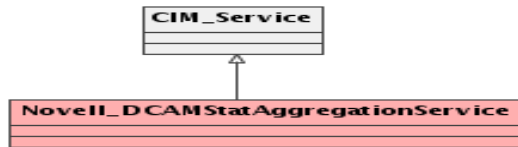
- **Novell_DCAMStatGatheringService**
 - Represents the service that gathers statistics related data from the local and remote computer systems
 - Ganglia GMOND gathering service
- **Novell_DCAMStatDef**
 - Represents a statistics definition as viewed by the gathering service. This includes elements such as Name, Sampling Interval, Send Threshold, Data Type, etc.
 - Single statistic gathered by a Ganglia GMOND metric module



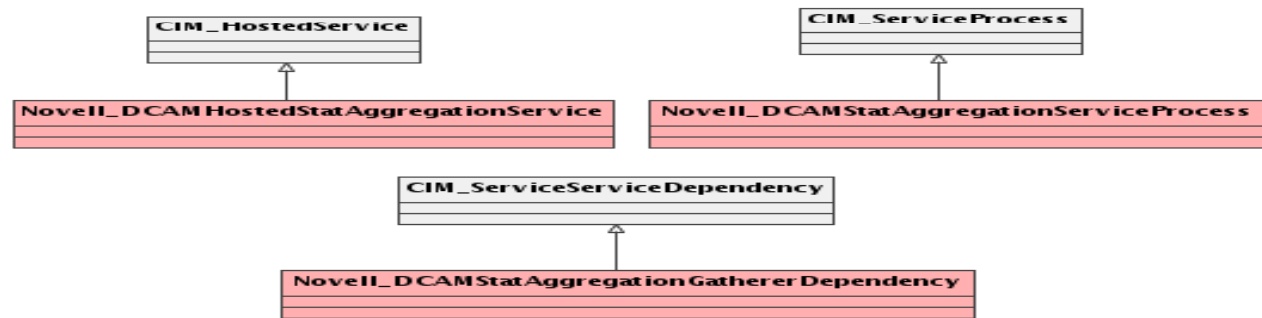
Gmetad – Statistic Aggregation Service



Class Inheritance Hierarchy



Association Inheritance Hierarchy





Gmetad – Statistics Aggregation Classes

- `Novell_DCAMStatAggregationService`
 - Represents the service that collects statistical data from various monitored systems
 - Ganglia GMETAD collection service

Demo – Accessing Statistical Values in CIM

Extending the System



Gmond Pluggable Metric Modules

- Extends the available metrics that can be gathered by Gmond
- Provided as dynamically loadable modules
- Configured through the `gmond.conf`
- Scheduled through Gmond rather than an external scheduler
- Module development is similar to an Apache module
- Able to produce multiple metrics from a single module



Gmond Module Development

- Three callback interfaces

- Init `int (*ex_metric_init)(apr_pool_t *p);`
- Clean up `void (*ex_metric_cleanup)(void);`
- Metric gathering handler `g_val_t (*ex_metric_handler)(int metric_index);`

- Metric definition structure

```
mmodule example_module =  
{  
    STD_MMODULE_STUFF, // Internal module definition  
    ex_metric_init,    // Metric init callback function  
    ex_metric_cleanup, // Metric cleanup callback function  
    ex_metric_info,    // Metric info data structure  
    ex_metric_handler, // Metric handler  
};
```

Gmond Example Module

```

mmodule example_module;

static int ex_metric_init(apr_pool_t *p)
{
    apr_array_header_t *list_params =
        example_module.module_params_list
    srand(time(NULL)%99);
    return 0;
}

static void ex_metric_cleanup ( void )
{
}

static g_val_t ex_metric_handler ( int
                                   metric_index )
{
    g_val_t val;
    switch (metric_index) {
        case 0:
            val.int32 = rand()%99;
            return val;
        case 1:
            val.int32 = 50;
            return val;
    }
    /* default case */
    val.int32 = 0;
    return val;
}

```

```

static const Ganglia_25metric
ex_metric_info[] =
{
    {0, "Random_Numbers", 90,
    GANGLIA_VALUE_UNSIGNED_INT, "s", "both",
    "%u", UDP_HEADER_SIZE+8,
    "Example module metric (random numbers)"},
    {0, "Constant_Number", 90,
    GANGLIA_VALUE_UNSIGNED_INT, "Num", "zero",
    "%hu", UDP_HEADER_SIZE+8,
    "Example module metric(constant number)"},
    {0, NULL}
};

mmodule example_module =
{
    STD_MMODULE_STUFF,
    ex_metric_init,
    ex_metric_cleanup,
    ex_metric_info,
    ex_metric_handler,
};

```



Gmond Example Module Configuration

```
modules {
  module {
    name = "example_module"
    path =
"/usr/lib/ganglia/modexample.so"
    param RandomMax {
      Value = 75
    }
    Param ConstantValue {
      Value = 25
    }
  }
}
```

```
/* Define Collection Groups */
collection_group {
  collect_every = 10
  time_threshold = 50
  metric {
    name = "Random_Numbers"
    value_threshold = 30.0
  }
}

collection_group {
  collect_once = yes
  time_threshold = 20
  metric {
    name = "Constant_Number"
  }
}
```



Gmond Python Module Development

- Extends the available metrics that can be gathered by Gmond
- Configured through the Gmond configuration file
- Python module interface is similar to the C module interface
- Ability to save state within the script vs. a persistent data store
- Larger footprint but easier to implement new metrics



Gmond Python Module Development

- Three mandatory functions
 - `metric_init()`
 - Called once at module initialization time
 - Must return a metric description dictionary or list of dictionaries
 - Any other module initialization can also take place here
 - `metric_handler()` – may have multiple handlers
 - Metric gathering handler
 - Must return a single data value of the same type as specified in the metric description dictionary returned from the `metric_init()` function
 - `metric_cleanup()`
 - Called once at module termination time
 - Does not return a value



Gmond Python Module Development

- Metric definition data dictionary or dictionary list, must be returned from the `metric_init()` function

```
d = {'name': '<your_metric_name>',  
     'call_back': <call_back function>,  
     'time_max': int(<your_time_max>),  
     'value_type': '<string | uint | float | double>',  
     'units': '<your_units>',  
     'slope': '<zero | positive | negative | both>',  
     'format': '<your_format>',  
     'description': '<your_description>'}
```

- Can be a single dictionary or a list of dictionaries
- Must be returned from the `metric_init()` function



Gmond Python Module Development

```
Curve_Max = 15
v = int(1)
inc = int(1)
count = 0

def metric_init(params):
    Global Curve_Max

    if 'CurveMax' in params:
        Curve_Max = int(params['CurveMax'])

    d = {'name': 'Curve_Metric',
         'call_back': curve_handler,
         'time_max': int(60),
         'value_type': 'uint',
         'units': 'Seconds',
         'slope': 'both',
         'format': '%u',
         'description':
             'Shows a uniform curve'}

    return d
```

```
def curve_handler(name):
    global v, count, inc, Curve_Max
    v += inc
    count += 1
    if count > Curve_Max:
        count = 0
        inc = -inc

    return int(v)

def metric_cleanup():
    pass
```



Gmond Python Module Deployment

- Copy the .py file to a specific directory
 - The python modules directory is defined in the gmond.conf file
- Start Gmond using the `-m` parameter
 - Shows a list of all available metrics known to Gmond
 - The python based metric should be in the list
- Add the new python metric to a collection group just like any other metric
- Restart Gmond



Configuring Gmond for Python

- Must load the mod_python.so pluggable module

```
modules {
    module {
        name = "python_module"
        path = "/usr/lib/ganglia/modpython.so"
        params = "/usr/lib/ganglia/python_modules"
    }
}
```

- Must specify a python module path
 - The 'params' directive specifies the python module path
 - Mod_python will automatically load any .py module found in the specified path
- Recommend including the python metric module .pyconf files from within the same .conf file that loads the python support module
 - Include ('/etc/ganglia/conf.d/*.pyconf')

Questions