Ganglia를 이용한 이기종 클라우드 모니터링 시스템 구축

곽재혁

한국과학기술정보연구원

목 차

표 목 차

그 림 목 차

제 1 장 서론

클라우드 컴퓨팅에서 가장 기본적인 기능이면서 중요한 이슈 중의 하나는 모니터링이라고 볼 수 있다. 특히, 클라우드 환경에서는 사용자에게 가상화된 자원을 제공하는 경우가 일반적이므로 가상화된 컴퓨팅 자원이나 스토리지 공간에 대한 실시간적인 모니터링이 특히 중요하다.

아울러, 클라우드 컴퓨팅이 일반화되면서 다양한 운영체제 플랫폼별로 클라우드 컴퓨팅의 핵심 기능 요소 지원이 기본적으로 탑재되기 시작했다. 특히, 가상화 기술은 이미 리눅스, 솔라리스, 윈도우 등을 포함하여 다양한 플랫폼 상에서 가능하게 되었으며, 비가상화 환경과 비교하여 성능상으로도 크게 차이가 없을만큼 기술적으로 성숙된 단계에 이르렀다.

과학 응용 연구자의 운영체제 플랫폼 요구 사항은 연구 분야에 따라 다양할 수 있으며 클라우드 컴퓨팅이 본격적으로 과학 응용 커뮤니티에서 활용되기 위해서는 이기종 클라우드 환경이 요구될 것으로 예상된다. 이를 위해서는 이기종 자원 환경에서 이용가능한 모니터링 시스템 구축 선행되어야 하는데, 본 보고서에서는 Ganglia를 이용하여 이기종 클라우드 모니터링 시스템을 구축하기 위한 방법을 설명한다.

Ganglia는 클러스터나 그리드 모니터링 시스템으로 수년간 개발되어온 확장성있는 분산 모니터링 시스템으로 오픈 소스 프로젝트이며 GNU/Linux, AIX, HP-UX, IRIX 등의 다양한 운영체제 플랫폼을 기본적으로 지원한다. 또한, 모니터링 요소에 대한 확장이 용이하기 때문에 클라우드 환경에 특화된 모니터링 요소를 추가하는 것도 가능하다. 본 보고서에서는 실제로 Ganglia의 gmetric을 이용하여 모니터링 요소를 확장하여 사이언스 클라우드 모니터링 시스템을 구축한 사례를 소개하였다.

제 2 장 설치

다음 각 절에서는 운영체제 플랫폼별로 Ganglia를 설치하기 위한 방법을 설명한다. Ganglia는 다양한 운영체제 상에서 설치될 수 있지만, 각 운영체제별로 설치시에 문제점이 다르게 나타난다. 여기서는 각 운영체제별로 Ganglia 설치시에 발생할 수 있는 문제점과 이에 대한 해결책도 함께 제시한다. 참고로 여기서 사용된 Ganglia는 2.5.5 버전이며 http://ganglia.sourceforge.net에서 다운로드 받을 수 있다.

제 1 절 GNU/Linux

Ganglia를 설치하기 위해서는 먼저 RRDtool 라이브러리가 설치되어야 한다. RRDtool 라이브러리는 네트웍 대역폭, 서버의 평균 부하 등과 같은 시간대별 데이터를 고속으로 저장하고 그래프로 표현할 수 있는 오픈 소스 라이브러리이다. RRDtool은 http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/에서 소스를 다운로드받은 수 있으며, 1.0.46 버전을 사용하였다.

RRDtool을 설치하기 위해서는 다음과 같이 실행한다.

```
tar xvzf rrdtool-1.0.46.tar.gz
cd rrdtool-1.0.46
./configure --prefix=/usr/local/rrdtool-1.0.46
make
make install
```

이제 Ganglia를 설치하기 위해서 다음과 같이 실행한다. configure시에 RRDtool에 대한 설치 경로를 필요로 한다.

```
tar xvzf ganglia-monitor-core-2.5.5.tar.gz
cd ganglia-monitor-core-2.5.5
./configure CFLAGS="-I/usr/local/rrdtool-1.0.46/include"
CPPFLAGS="-I/usr/local/rrdtool-1.0.46/include"
LDFLAGS="-L/usr/local/rrdtool-1.0.46/lib" --prefix=/usr/local/ganglia-2.5.5
--with-gmetad --enable-gexec
make
make install
```

제 2 절 AIX

먼저 RRDtool 라이브러리를 설치하기 위해서 다음과 같이 실행한다.

```
gunzip –c rrdtool-1.0.46.tar.gz | tar –xvf –
cd rrdtool-1.0.46
./configure --prefix=/usr/local/rrdtool-1.0.46
make
```

Ganglia를 설치하기 위해서 다음과 같이 실행한다. configure시에 --disable-shared --enable-static옵션을 줄 필요가 있다.

```
gunzip –c ganglia-monitor-core-2.5.5.tar.gz | tar –xvf -
cd ganglia-monitor-core-2.5.5
./configure CFLAGS="-I/usr/local/rrdtool-1.0.46/include"
CPPFLAGS="-I/usr/local/rrdtool-1.0.46/include"
LDFLAGS="-L/usr/local/rrdtool-1.0.46/lib" --prefix=/usr/local/ganglia-2.5.5
--with-gmetad --enable-gexec --disable-shared --enable-static
make
```

소스를 컴파일하는 과정에서 다음과 같은 에러가 발생할 수 있다.

```
source='getopt1.c' object='getopt1.o' libtool=no ₩
depfile='.deps/getopt1.Po' tmpdepfile='.deps/getopt1.TPo' ₩
depmode=aix /bin/sh ../config/depcomp ₩
cc -DHAVE_CONFIG_H -I. -I. -I.. -I/usr/local/rrdtool-1.0.46/include -I.. -I.
-I./dnet -I/usr/local/rrdtool-1.0.46/include -Wall -D_ALL_SOURCE -DAIX
-DHAVE_PMAPI -c `test -f 'getopt1.c' || echo './'`getopt1.c
cc: 1501-210 command option Wall contains an incorrect subargument
```

위와 같은 에러가 발생시에 configure파일을 다음과 같이 수정한다.

```
vi configure
# CFLAGS="$CFLAGS -Wall"
CFLAGS="$CFLAGS"
```

configure를 다시 실행한 후에 소스를 다시 컴파일하는 과정에서 다음과 같은 에러가 발생할 수 있다.

```
source='server.c' object='server.o' libtool=no ₩
depfile='.deps/server.Po' tmpdepfile='.deps/server.TPo' ₩
depmode=aix /bin/sh ../config/depcomp ₩
cc -DHAVE_CONFIG_H -I. -I. -I..    -I/usr/local/rrdtool-1.0.46/include -O0 -I../lib
-I../gmond        -I/usr/local/rrdtool-1.0.46/include        -D_ALL_SOURCE        -DAIX
-DHAVE_PMAPI -c `test -f 'server.c' || echo './'`server.c
cc: 1501-216 command option 0 is not recognized - passed to ld
"gmetad.h", line 106.9: 1506-236 (W) Macro name FRAMESIZE has been redefined.
"gmetad.h", line 106.9: 1506-358 (I) "FRAMESIZE" is defined on line 253 of
/usr/include/sys/mstsave.h.
"server.c", line 22.1: 1506-277 (S) Syntax error: possible missing ';' or ','?
"server.c", line 20.8: 1506-485 (S) Parameter declaration list is incompatible with
declarator for inline.
"server.c", line 27.14: 1506-276 (S) Syntax error: possible missing '{'?
"server.c", line 31.10: 1506-045 (S) Undeclared identifier ap.
"server.c", line 35.20: 1506-276 (S) Syntax error: possible missing identifier?
"server.c", line 35.28: 1506-335 (S) Parameter identifier list contains multiple
occurrences of buf.
"server.c", line 35.4: 1506-273 (E) Missing type in declaration of vsnprintf.
"server.c", line 35.4: 1506-282 (S) The type of the parameters must be specified in
a prototype.
"server.c", line 35.41: 1506-275 (S) Unexpected text ')' encountered.
"server.c", line 37.4: 1506-273 (E) Missing type in declaration of len.
"server.c", line 37.17: 1506-045 (S) Undeclared identifier buf.
"server.c", line 37.10: 1506-221 (S) Initializer must be a valid constant expression.
"server.c", line 39.4: 1506-046 (S) Syntax error.
```

위와 같은 에러가 발생시에 gmetad/Makefile파일을 다음과 같이 수정한다.

```
vi gmetad/Makefile
# AM_CFLAGS = -O0 -I../lib -I../gmond
```

```
AM_CFLAGS = -I../lib -I../gmond
```

또한, gmetad/server.c파일을 다음과 같이 수정한다.

```
vi gmetad/server.c
/* static inline int */
static int
xml_print( client_t *client, const char *fmt, ... )
```

소스를 다시 컴파일하는 과정에서 다음과 같은 에러가 발생할 수 있다.

```
source='rrd_helpers.c' object='rrd_helpers.o' libtool=no ₩
depfile='.deps/rrd_helpers.Po' tmpdepfile='.deps/rrd_helpers.TPo' ₩
depmode=aix /bin/sh ../config/depcomp ₩
cc -DHAVE_CONFIG_H -I. -I. -I.. -I/usr/local/rrdtool-1.0.46/include -I../lib
-I../gmond -I/usr/local/rrdtool-1.0.46/include -D_ALL_SOURCE -DAIX
-DHAVE_PMAPI -c `test -f 'rrd_helpers.c' || echo './'`rrd_helpers.c
"./gmetad.h", line 106.9: 1506-236 (W) Macro name FRAMESIZE has been
redefined.
"./gmetad.h", line 106.9: 1506-358 (I) "FRAMESIZE" is defined on line 253 of
/usr/include/sys/mstsave.h.
"rrd_helpers.c", line 20.13: 1506-160 (S) Object inline cannot be declared as type
void.
"rrd_helpers.c", line 20.13: 1506-166 (S) Definition of function inline requires
parentheses.
"rrd_helpers.c", line 21.1: 1506-276 (S) Syntax error: possible missing '{'?
"rrd_helpers.c", line 23.17: 1506-045 (S) Undeclared identifier dir.
```

위와 같은 에러가 발생시에 gmetad/rrd_helpers.c파일을 다음과 같이 수정한다.

```
vi gmetad/rrd_helpers.c
/* static void inline */
static void
my_mkdir ( const char *dir )
```

소스를 다시 컴파일하는 과정에서 다음과 같은 에러가 발생할 수 있다.

```
source='server.c' object='server.o' libtool=no ₩
depfile='.deps/server.Po' tmpdepfile='.deps/server.TPo' ₩
depmode=aix /bin/sh ../config/depcomp ₩
cc -DHAVE_CONFIG_H -I. -I. -I.. -I/usr/local/rrdtool-1.0.46/include -I../lib
-I../lib/dnet -I/usr/local/rrdtool-1.0.46/include -D_ALL_SOURCE -DAIX
-DHAVE_PMAPI -c `test -f 'server.c' || echo './'`server.c
"node_data_t.h", line 23.9: 1506-236 (W) Macro name FRAMESIZE has been
redefined.
"node_data_t.h", line 23.9: 1506-358 (I) "FRAMESIZE" is defined on line 253 of
/usr/include/sys/mstsave.h.
"server.c", line 33.1: 1506-277 (S) Syntax error: possible missing ';' or ','?
"server.c", line 31.8: 1506-485 (S) Parameter declaration list is incompatible with
declarator for inline.
"server.c", line 38.14: 1506-276 (S) Syntax error: possible missing '{'?
"server.c", line 42.10: 1506-045 (S) Undeclared identifier ap.
"server.c", line 46.20: 1506-276 (S) Syntax error: possible missing identifier?
"server.c", line 46.28: 1506-335 (S) Parameter identifier list contains multiple
occurrences of buf.
"server.c", line 46.4: 1506-273 (E) Missing type in declaration of vsnprintf.
"server.c", line 46.4: 1506-282 (S) The type of the parameters must be specified in
a prototype.
"server.c", line 46.41: 1506-275 (S) Unexpected text ')' encountered.
"server.c", line 48.4: 1506-273 (E) Missing type in declaration of len.
"server.c", line 48.17: 1506-045 (S) Undeclared identifier buf.
"server.c", line 48.10: 1506-221 (S) Initializer must be a valid constant expression.
"server.c", line 50.4: 1506-046 (S) Syntax error.
```

위와 같은 에러가 발생시에 gmond/server.c파일을 다음과 같이 수정한다.

```
vi gmond/server.c
/* static inline int */
static int
buffrd_print( client_t *client, const char *fmt, ... )
```

소스를 다시 컴파일하는 과정에서 다음과 같은 에러가 발생할 수 있다.

```
source='cleanup.c' object='cleanup.o' libtool=no ₩
depfile='.deps/cleanup.Po' tmpdepfile='.deps/cleanup.TPo' ₩
depmode=aix /bin/sh ../config/depcomp ₩
cc -DHAVE_CONFIG_H -I. -I.-I.. -I/usr/local/rrdtool-1.0.46/include -I../lib
-I../lib/dnet -I/usr/local/rrdtool-1.0.46/include -D_ALL_SOURCE -DAIX
-DHAVE_PMAPI -c `test -f 'cleanup.c' || echo './'`cleanup.c
"node_data_t.h", line 23.9: 1506-236 (W) Macro name FRAMESIZE has been
redefined.
"node_data_t.h", line 23.9: 1506-358 (I) "FRAMESIZE" is defined on line 253 of
/usr/include/sys/mstsave.h.
"cleanup.c", line 145.7: 1506-046 (S) Syntax error.
```

위와 같은 에러 발생시에 gmond/cleanup.c파일을 다음과 같이 수정한다.

```
vi gmond/cleanup.c
// report_stats();
/* report_stats(); */
```

제 3 절 HP-UX

먼저 rrdtool 라이브러리를 설치하기 위해서 다음과 같이 실행한다.

```
gunzip –c rrdtool-1.0.46.tar.gz | tar –xvf –
cd rrdtool-1.0.46
./configure --prefix=/usr/local/rrdtool-1.0.46
make
```

소스를 컴파일하는 과정에서 다음과 같은 에러가 발생할 수 있다.

```
LD_RUN_PATH="" ld  -b +vnocompatwarnings -L/usr/local/lib RRDs.o -L../src/.libs/
-lrrd_private -lm -o blib/arch/auto/RRDs/RRDs.sl
ld: Invalid loader fixup in text space needed in output file for symbol "$00000124"
in input
```

```
file "../src/.libs//librrd_private.a(rrd_graph.o)"
make[3]: *** [blib/arch/auto/RRDs/RRDs.sl] Error 1
```

위와 같은 에러 발생시에 configure시 --with-pic옵션을 사용하여 다음과 같이 실행한다.

```
./configure --prefix=/usr/local/rrdtool-1.0.46 --with-pic
```

Ganglia를 설치하기 위해서 다음과 같이 실행한다.

```
gunzip –c ganglia-monitor-core-2.5.5.tar.gz | tar –xvf -
cd ganglia-monitor-core-2.5.5
./configure CFLAGS="-I/usr/local/rrdtool-1.0.46/include"
CPPFLAGS="-I/usr/local/rrdtool-1.0.46/include"
LDFLAGS="-L/usr/local/rrdtool-1.0.46/lib" --prefix=/usr/local/ganglia-2.5.5
--with-gmetad --enable-gexec
make
```

소스를 컴파일하는 과정에서 다음과 같은 에러가 발생할 수 있다.

```
cc -DHAVE_CONFIG_H -I. -I. -I..    -I/usr/local/rrdtool-1.0.46/include -O0 -I../lib
-I../gmond -I/usr/local/rrdtool-1.0.46/include -Wall -D_HPUX_SOURCE -c `test -f
'server.c' || echo './'`server.c
cc: "server.c", line 20: error 1000: Unexpected symbol: "int".
```

위와 같은 에러 발생시에 gmetad/server.c파일을 다음과 같이 수정한다.

```
vi gmetad/server.c
//static inline int
static int
xml_print( client_t *client, const char *fmt, ... )
```

소스를 다시 컴파일하는 과정에서 다음과 같은 에러가 발생할 수 있다.

```
cc -DHAVE_CONFIG_H -I. -I. -I..    -I/usr/local/rrdtool-1.0.46/include-O0 -I../lib
-I../gmond -I/usr/local/rrdtool-1.0.46/include -Wall -D_HPUX_SOURCE -c `test -f
'rrd_helpers.c' || echo './'`rrd_helpers.c
cc: "rrd_helpers.c", line 21: error 1000: Unexpected symbol: "my_mkdir".
```

위와 같은 에러 발생시에 gmetad/rrd_helpers.c파일을 다음과 같이 수정한다.

```
vi gmetad/rrd_helpers.c
//static void inline
static void
my_mkdir ( const char *dir )
```

소스를 다시 컴파일하는 과정에서 다음과 같은 에러가 발생할 수 있다.

```
cc -DHAVE_CONFIG_H -I. -I. -I.. -I/usr/local/rrdtool-1.0.46/include -I../lib
-I../lib/dnet -I/usr/local/rrdtool-1.0.46/include -Wall -D_HPUX_SOURCE -c `test -f
'server.c' || echo './'`server.c
cc: "server.c", line 31: error 1000: Unexpected symbol: "int".
```

위와 같은 에러 발생시에 gmond/server.c파일을 다음과 같이 수정한다.

```
vi gmond/server.c
//static inline int
static int
buffrd_print( client_t *client, const char *fmt, ... )
```

제 4 절 IRIX

먼저 RRDtool 라이브러리를 설치하기 위해서 다음과 같이 실행한다.

```
gunzip –c rrdtool-1.0.46.tar.gz | tar –xvf –
cd rrdtool-1.0.46
./configure --prefix=/usr/local/rrdtool-1.0.46
make
make install
```

Ganglia를 설치하기 위해서 다음과 같이 실행한다.

```
gunzip –c ganglia-monitor-core-2.5.5.tar.gz | tar –xvf –
cd ganglia-monitor-core-2.5.5
./configure CFLAGS="-I/usr/local/rrdtool-1.0.46/include"
```

```
CPPFLAGS="-I/usr/local/rrdtool-1.0.46/include"
LDFLAGS="-L/usr/local/rrdtool-1.0.46/lib" --prefix=/usr/local/ganglia-2.5.5
--with-gmetad --enable-gexec
make
```

소스를 컴파일하는 과정에서 다음과 같은 에러가 발생할 수 있다.

```
In file included from /usr/include/stropts.h:38,
                 from gangliaconf.h:54,
                 from daemon_inetd.c:5:
/usr/include/sys/stropts.h:237: parse error before "t_uscalar_t"
/usr/include/sys/stropts.h:261: parse error before "t_uscalar_t"
/usr/include/sys/stropts.h:267: parse error before '}' token
```

위와 같은 에러 발생시에 lib/gangliaconf.h파일을 다음과 같이 수정한다.

```
vi lib/gangliaconf.h
//#ifdef HAVE_STROPTS_H
//#include <stropts.h>
//#endif

#ifdef HAVE_STROPTS_H
#undef _XOPEN5
#include <stropts.h>
#endif
```

제 3 장 설정

제 1 절 초기화 스크립트 작성

Ganglia는 GNU/Linux에서 사용되는 gmond와 gmetad에 대한 초기화 스크립트를 제공하며, 소스 디렉토리로부터 복사하기 위해서 다음과 같이 실행한다.

```
cp gmond/gmond.init /etc/rc.d/init.d/gmond
cp gmetad/gmetad.init /etc/rc.d/init.d/gmetad
chkconfig --add gmond
chkconfig --add gmetad
```

gmond가 설치된 위치를 수정하기 위해서 gmond를 다음과 같이 수정한다.

```
vi /etc/rc.d/init.d/gmond
GMOND=/usr/local/ganglia/sbin/gmond
```

gmetad가 설치된 위치를 수정하기 위해서 gmetad를 다음과 같이 수정한다.

```
vi /etc/rc.d/init.d/gmetad
GMETAD=/usr/local/ganglia/sbin/gmetad
```

제 2 절 gmond 설정

gmond는 Ganglia 모니터링 데몬으로 클러스터 멀티캐스트 채널에서 리스닝하고 있으며 데이터를 메모리에 저장하고 있다가 요청시 오면 클러스터의 상태를 XML형태로 출력하는 역할을 담당한다.

Ganglia는 gmond를 위한 설정 파일을 제공하며 소스 디렉토리로부터 복사하기 위해서 다음과 같이 실행한다.

```
cp gmond/gmond.conf /etc
```

다음은 GNU/Linux에 설치된 gmond.conf 설정의 한 예이다.

```
# $Id: gmond.conf,v 1.2 2002/09/19 00:37:18 sacerdoti Exp $
# 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   "venus"
#
# The owner of this cluster. Represents an administrative
# domain. The pair name/owner should be unique for all clusters
# in the world.
# default: "unspecified"
 owner "gridcenter.or.kr"
#
# The latitude and longitude GPS coordinates of this cluster on earth.
# Specified to 1 mile accuracy with two decimal places per axis in Decimal
# DMS format: "N61.18 W130.50".
# default: "unspecified"
# latlong "N32.87 W117.22"
#
# The URL for more information on the Cluster. Intended to give purpose,
# owner, administration, and account details for this cluster.
# default: "unspecified"
# url "http://www.mycluster.edu/"
#
# The location of this host in the cluster. Given as a 3D coordinate:
# "Rack,Rank,Plane" that corresponds to a Euclidean coordinate "x,y,z".
# default: "unspecified"
# location "0,0,0"
#
# 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      20649
#
# The multicast interface for gmond to send/receive data on
# default: the kernel decides based on routing configuration
 mcast_if   eth0
#
# 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       20649
#
# 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.
# Can be specified with either hostnames or IP addresses.
# default: none
 trusted_hosts 150.183.249.12
#
```

```
# The number of nodes in your cluster.  This value is used in the
# creation of the cluster hash.
# default: 1024
 num_nodes  64
#
# 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
```

제 3 절 gmetad 설정

gmetad는 Ganglia 메타 데몬으로 다수의 gmond나 gmetad로부터 정보를 수집하고, 이를 로컬 round-robin 데이터베이스에 저장하며, 모든 데이터 소스를 취합하여 XML 형태로 제공하는 역할을 담당한다.

Ganglia는 gmetad를 위한 설정 파일을 제공하며, 소스 디렉토리로부터 복사하기 위해서 다음과 같이 실행한다.

```
cp gmetad/gmetad.conf /etc
```

다음은 GNU/Linux의 gmetad.conf의 한 예이다.

```
# This is an example of a Ganglia Meta Daemon configuration file
#                    http://ganglia.sourceforge.net/
#
# $Id: gmetad.conf,v 1.10 2003/08/06 23:11:33 sacerdoti Exp $
#
#-------------------------------------------------------------------
# Setting the debug_level to 1 will keep daemon in the forground and
# show only error messages. Setting this value higher than 1 will make
# gmetad output debugging information and stay in the foreground.
# default: 0
# debug_level 10
#
```

```
#------------------------------------------------------------------
# What to monitor. The most important section of this file.
#
# The data_source tag specifies either a cluster or a grid to
# monitor. If we detect the source is a cluster, we will maintain a complete
# set of RRD databases for it, which can be used to create historical
# graphs of the metrics. If the source is a grid (it comes from another gmetad),
# we will only maintain summary RRDs for it.
#
# Format:
# data_source "my cluster" [polling interval] address1:port addreses2:port ...
#
# The keyword 'data_source' must immediately be followed by a unique
# string which identifies the source, then an optional polling interval in
# seconds. The source will be polled at this interval on average.
# If the polling interval is omitted, 15sec is asssumed.
#
# A list of machines which service the data source follows, in the
# format ip:port, or name:port. If a port is not specified then 8649
# (the default gmond port) is assumed.
# default: There is no default value
#
# 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

data_source "venus" 150.183.249.12:20649

#
#------------------------------------------------------------------
# Scalability mode. If on, we summarize over downstream grids, and respect
# authority tags. If off, we take on 2.5.0-era behavior: we do not wrap our output
# in <GRID></GRID> tags, we ignore all <GRID> tags we see, and always assume
```

```
# we are the "authority" on data source feeds. This approach does not scale to
# large groups of clusters, but is provided for backwards compatibility.
# default: on
# scalable off
#
#------------------------------------------------------------------------
# The name of this Grid. All the data sources above will be wrapped in a GRID
# tag with this name.
# default: Unspecified
 gridname "venus"
#
#------------------------------------------------------------------------
# The authority URL for this grid. Used by other gmetads to locate graphs
# for our data sources. Generally points to a ganglia/
# website on this machine.
# default: "http://hostname/ganglia/",
#    where hostname is the name of this machine, as defined by gethostname().
 authority "http://150.183.249.12:24580/ganglia/"
#
#------------------------------------------------------------------------
# List of machines this gmetad will share XML with. Localhost
# is always trusted.
# default: There is no default value
 trusted_hosts 150.183.249.12 150.183.249.8
#
#------------------------------------------------------------------------
# If you want any host which connects to the gmetad XML to receive
# data, then set this value to "on"
# default: off
# all_trusted on
#
#------------------------------------------------------------------------
# If you don't want gmetad to setuid then set this to off
```

```
# default: on
# setuid off
#
#------------------------------------------------------------------
# User gmetad will setuid to (defaults to "nobody")
# default: "nobody"
# setuid_username "nobody"
#
#------------------------------------------------------------------
# The port gmetad will answer requests for XML
# default: 8651
 xml_port 20651
#
#------------------------------------------------------------------
# The port gmetad will answer queries for XML. This facility allows
# simple subtree and summation views of the XML tree.
# default: 8652
 interactive_port 20652
#
#------------------------------------------------------------------
# The number of threads answering XML requests
# default: 4
# server_threads 10
#
#------------------------------------------------------------------
# Where gmetad stores its round-robin databases
# default: "/var/lib/ganglia/rrds"
# rrd_rootdir "/some/other/place"
```

제 4 장 실행

Ganglia는 기본적으로 클러스터 단위로 모니터링 시스템을 운영한다. 따라서 클러스터의 마스터 노드에는 gmetad와 gmond가 실행되어야 하고, 슬레이브 노드에는 gmond가 실행되어야 한다.

먼저 gmetad가 rrd파일을 저장하는 디렉토리를 생성하기 위해서 다음과 같이 실행한다.

```
mkdir –p /var/lib/ganglia/rrds
chown –R nobody.nobody /var/lib/ganglia
```

마스터 노드에서 Ganglia를 시작하기 위해서 다음과 같이 실행한다.

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

슬레이브 노드에서 Ganglia를 시작하기 위해서 다음과 같이 실행한다. 여기서 cexecs는 클러스터의 각 슬레이브 노드별로 동일한 작업을 반복할 수 있게 하는 유틸리티이다.

```
cexecs /etc/rc.d/init.d/gmond restart
```

제 5 장 웹환경 설치

Ganglia 웹환경은 PHP로 작성되었으며 ganglia를 통해서 수집된 정보를 동적으로 보여준다. 사용자별로 원하는 형태의 데이터만을 보여줄 수 있으며, 시간별, 일별, 주별, 월별, 년별로 데이터를 보여주는 것도 가능하다.

Ganglia 웹환경의 소스 압축을 풀기 위해서 다음과 같이 실행한다.

```
cd /var/www/
tar xvzf ganglia-webfrontend-2.5.5.tar.gz
```

Ganglia 웹환경을 설정하기 위해서 conf.php를 다음과 같이 수정한다.

```
cd ganglia-webfrontend-2.5.5
vi conf.php
# The high-performance gmetad.
$gmetad_root = "/usr/local/ganglia/var/lib/ganglia";
```

```
$rrds = "$gmetad_root/rrds";

# Leave this alone if rrdtool is installed in $gmetad_root,
# otherwise, change it if it is installed elsewhere (like /usr/bin)
define("RRDTOOL", "/usr/local/rrdtool/bin/rrdtool");

#
# If you want to grab data from a different ganglia source specify it here.
# Although, it would be strange to alter the IP since the Round-Robin
# databases need to be local to be read.
#
$ganglia_ip = "150.183.249.8";
$ganglia_port = 20652;
```

Ganglia 웹환경은 별도의 웹서버 서버에 설치되어 운영될 수 있으며, 여러 ganglia 소스로부터의 정보를 취합하는 별도의 gmetad에 의존한다. Ganglia 웹환경이 설치된 웹서버 노드에는 RRDtool이 설치되어 있어야 하며 gmetad가 실행되고 있어야 한다.

다음은 Ganglia 웹환경이 설치된 웹서버 노드의 gmetad.conf의 한 예이다.

```
# This is an example of a Ganglia Meta Daemon configuration file
#                    http://ganglia.sourceforge.net/
#
# $Id: gmetad.conf,v 1.10 2003/08/06 23:11:33sacerdoti Exp $
#
#-------------------------------------------------------------------
# Setting the debug_level to 1 will keep daemon in the forground and
# show only error messages. Setting this value higher than 1 will make
# gmetad output debugging information and stay in the foreground.
# default: 0
# debug_level 10
#
#-------------------------------------------------------------------
# What to monitor. The most important section of this file.
#
```

```
# The data_source tag specifies either a cluster or a grid to
# monitor. If we detect the source is a cluster, we will maintain a complete
# set of RRD databases for it, which can be used to create historical
# graphs of the metrics. If the source is a grid (it comes from another gmetad),
# we will only maintain summary RRDs for it.
#
# Format:
# data_source "my cluster" [polling interval] address1:port addreses2:port ...
#
# The keyword 'data_source' must immediately be followed by a unique
# string which identifies the source, then an optional polling interval in
# seconds. The source will be polled at this interval on average.
# If the polling interval is omitted, 15sec is asssumed.
#
# A list of machines which service the data source follows, in the
# format ip:port, or name:port. If a port is not specified then 8649
# (the default gmond port) is assumed.
# default: There is no default value
#
# 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

#data_source "gw" 150.183.249.8:20649

data_source "venus" 150.183.249.12:20651
data_source "jupiter" 150.183.249.14:20651
data_source "sun" 150.183.249.10:20651
data_source "trinitas" 203.30.109.15:20651
data_source "supercom3" 141.223.113.3:20651

#
#------------------------------------------------------------------
```

```
# Scalability mode. If on, we summarize over downstream grids, and respect
# authority tags. If off, we take on 2.5.0-era behavior: we do not wrap our output
# in <GRID></GRID> tags, we ignore all <GRID> tags we see, and always assume
# we are the "authority" on data source feeds. This approach does not scale to
# large groups of clusters, but is provided for backwards compatibility.
# default: on
# scalable off
#
#------------------------------------------------------------------------
# The name of this Grid. All the data sources above will be wrapped in a GRID
# tag with this name.
# default: Unspecified
 gridname "MyCloud"
#
#------------------------------------------------------------------------
# The authority URL for this grid. Used by other gmetads to locate graphs
# for our data sources. Generally points to a ganglia/
# website on this machine.
# default: "http://hostname/ganglia/",
#    where hostname is the name of this machine, as defined by gethostname().
 authority "https://150.183.249.8:20443/ganglia/"
#
#------------------------------------------------------------------------
# List of machines this gmetad will share XML with. Localhost
# is always trusted.
# default: There is no default value
 trusted_hosts 150.183.249.8
#
#------------------------------------------------------------------------
# If you want any host which connects to the gmetad XML to receive
# data, then set this value to "on"
# default: off
# all_trusted on
```

```
#
#--------------------------------------------------------------------
# If you don't want gmetad to setuid then set this to off
# default: on
# setuid off
#
#--------------------------------------------------------------------
# User gmetad will setuid to (defaults to "nobody")
# default: "nobody"
 setuid_username "jhkwak"
#
#--------------------------------------------------------------------
# The port gmetad will answer requests for XML
# default: 8651
 xml_port 20651
#
#--------------------------------------------------------------------
# The port gmetad will answer queries for XML. This facility allows
# simple subtree and summation views of the XML tree.
# default: 8652
 interactive_port 20652
#
#--------------------------------------------------------------------
# The number of threads answering XML requests
# default: 4
# server_threads 10
#
#--------------------------------------------------------------------
# Where gmetad stores its round-robin databases
# default: "/var/lib/ganglia/rrds"
 rrd_rootdir "/usr/local/ganglia/var/lib/ganglia/rrds"
```

Ganglia 웹환경 설치 노드에서 gmetad를 시작하기 위해서 다음과 같이 실행한다.

```
cd /usr/local/ganglia
sbin/gmetad --conf='./gmetad.conf'
```

제 6 장 확장

Ganglia는 gmetric이라는 Ganglia 매트릭 클라이언트를 제공하여 ganglia가 커스텀 매트릭 정보를 수집할 수 있게 한다. gmetric을 통해서 실행된 매트릭 값은 클러스터 멀티캐스트 채널에서 리스닝하고 있는 gmond에 의해서 수집된다. 오픈소스 커뮤니티로부터 개발된 다양한 gmetric 커스텀 스크립트가 이용가능하며 http://ganglia.info/gmetric/에서 다운로드 받을 수 있다.

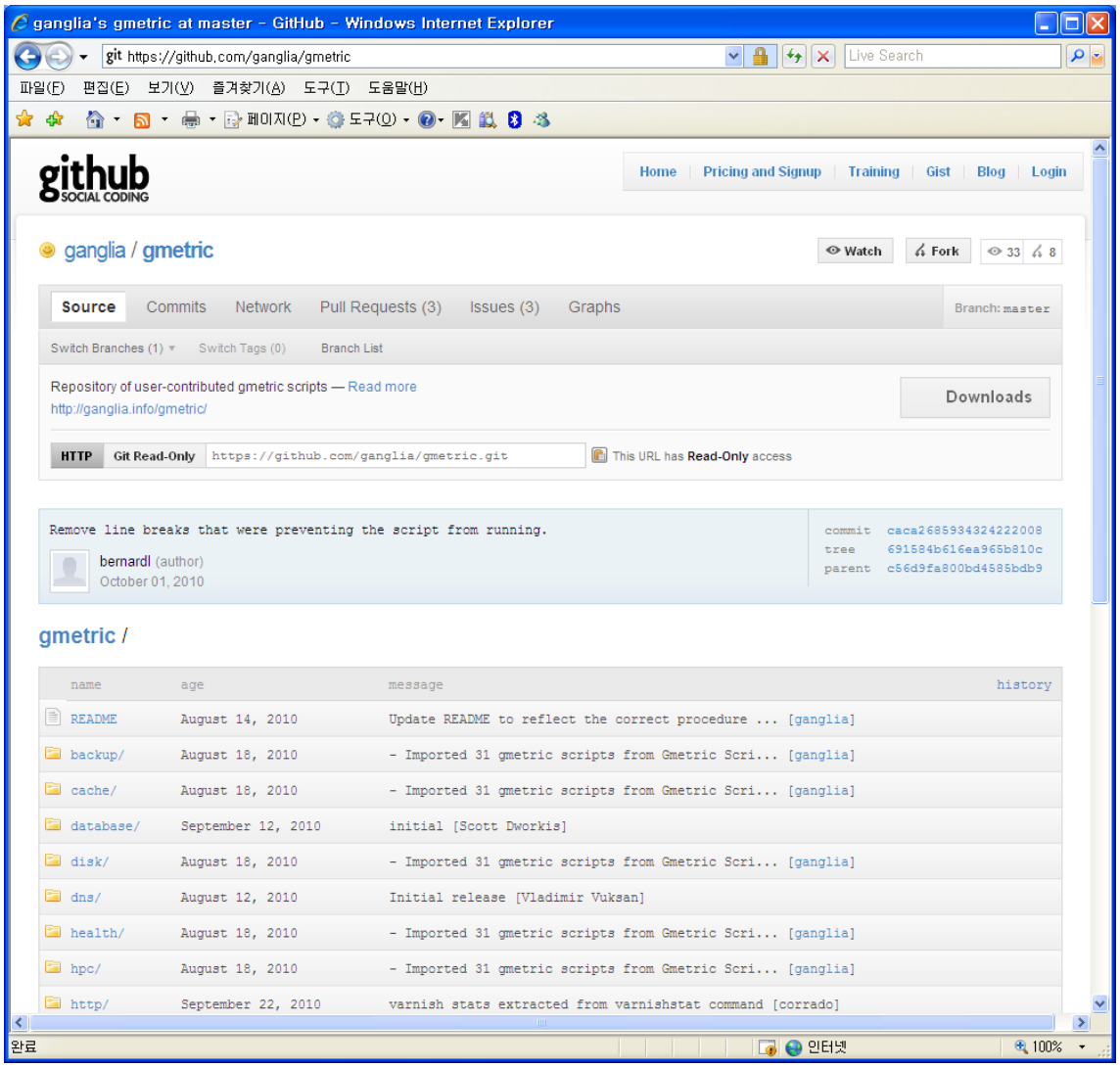

그림 1 gmetric 스크립트 저장소

gmetric은 다음과 같은 옵션 전달인자를 이용하여 커스텀 매트릭을 정의하고 ganglia에 정보를 전달할 수 있다.

표 1 gmetric 옵션 전달인자

옵션 전달인자	의미
-nSTRING --name=STRING	매트릭 이름
-vSTRING --value=STRING	매트릭 값
-tSTRING --type=STRING	매트릭 타입, 다음의 값 중 하나 string|int8|uint8|int16|uint16|int32|uint32|float|double
-uSTRING --units=STRING	메트릭 값 측정 단위, 예를 들면, Kilobytes, Celcius

다음은 gmetric을 이용하여 GNU/Linux NFS 클라이언트 통계 정보를 ganglia로 보내는 gmetric 커스텀 스크립트의 한 예이다. 아래 스크립트는 crond에 등록되어 주기적으로 실행되어야 한다.

```
#!/bin/bash
#
# Linux NFS Client statistics
#
# Report number of NFS client read, write and getattr calls since we were last called.
# (Use utility "nfsstat -c" to look at the same thing).
# Note: Uses temp files in /tmp
#

# GETATTR
if [ -f /tmp/nfsclientgetattr ]; then
        thisnfsgetattr=`cat /proc/net/rpc/nfs | tail -1 | awk '{printf "%s\n",$4}'`
        lastnfsgetattr=`cat /tmp/nfsclientgetattr`
        let "deltagetattr = thisnfsgetattr - lastnfsgetattr"
        # echo "delta getattr $deltagetattr"
        /usr/bin/gmetric -nnfsgetattr -v$deltagetattr -tuint16 -ucalls
fi

# READ
if [ -f /tmp/nfsclientread ]; then
```

```
        thisnfsread=`cat /proc/net/rpc/nfs | tail -1 | awk '{printf "%s\n",$9}'`
        lastnfsread=`cat /tmp/nfsclientread`
        let "deltaread = thisnfsread - lastnfsread"
        # echo "delta read $deltaread"
        /usr/bin/gmetric -nnfsread -v$deltaread -tuint16 -ucalls
fi

# WRITE
if [ -f /tmp/nfsclientwrite ]; then
        thisnfswrite=`cat /proc/net/rpc/nfs | tail -1 | awk '{printf "%s\n",$10}
'`
        lastnfswrite=`cat /tmp/nfsclientwrite`
        let "deltawrite = thisnfswrite - lastnfswrite"
        # echo "delta write $deltawrite"
        /usr/bin/gmetric -nnfswrite -v$deltawrite -tuint16 -ucalls
fi

# NFS Quality Assurance RATIO (nfsqaratio)
# If this value shrinks too much then perhaps an application
# program change introduced excessive GETATTR calls into production.
if [ "$deltagetattr" -ne 0 ];then
          let "nfsqaratio = (deltaread + deltawrite) / deltagetattr"
          /usr/bin/gmetric -nnfsqaratio -v$nfsqaratio -tuint16 -ucalls
fi

# Update the old values on disk for the next time around. (We ignore
# the fact that they have probably already changed while we made this
# calculation).
cat /proc/net/rpc/nfs | tail -1 | awk '{printf "%s\n",$9}'  > /tmp/nfsclientread
cat /proc/net/rpc/nfs | tail -1 | awk '{printf "%s\n",$10}' > /tmp/nfsclientwrite
cat /proc/net/rpc/nfs | tail -1 | awk '{printf "%s\n",$4}'  > /tmp/nfsclientgetattr
```

제 7 장 사례연구: 사이언스 클라우드 모니터링 시스템

본 장에서는 ganglia를 이용하여 구축된 사이언스 클라우드 모니터링 시스템 구축 사례를 소개한다. 사이언스 클라우드는 과학응용 연구자들에게 클라우드 기반의 가상화된 연구 환경을 제공하기 위한 목적으로 구축되고 있으며, 1차년도에는 오픈 소스 인프라스트럭처 클라우드 소프트웨어인 Eucalyptus(http://www.eucalyptus.org)를 이용하여 구축되었다.

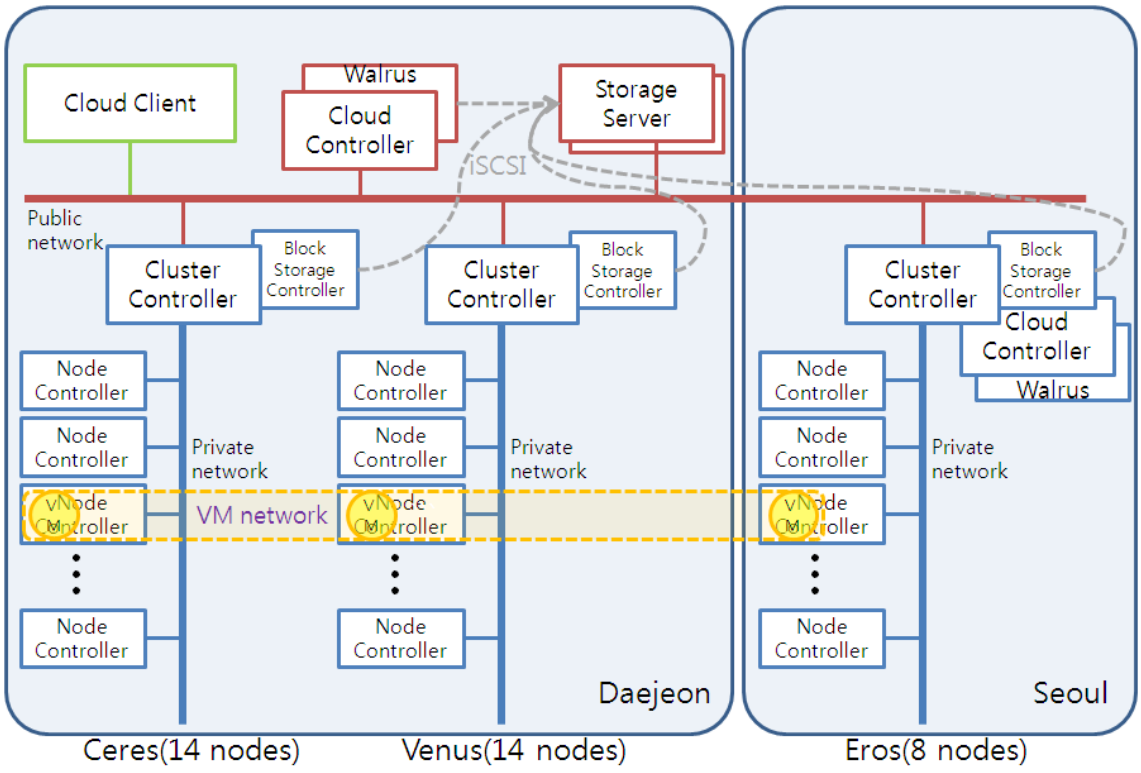

그림 2 사이언스 클라우드 구성도

사이언스 클라우드에서는 전체 물리 자원에 대한 모니터링 시스템으로서 ganglia를 사용하였으며 사용자에게 제공되는 가상화된 컴퓨팅 환경과 스토리지 공간을 모니터링하기 위해서 gmetric을 이용하여 ganglia 정보를 확장하였다. 다음은 Eucalyptus에서 제공하는 정보를 가공하고 gmetric을 통해서 ganglia가 읽어들일 수 있는 형태로 변환화는 스크립트이다.

```
#!/bin/sh
TYPE="node"
NC_STAT="/var/run/eucalyptus/nc-stats"
SC_STAT="/var/log/eucalyptus/sc-stats.log"
WALRUS_STAT="/var/log/eucalyptus/walrus-stats.log"
```

```
GMETRIC="`which gmetric 2> /dev/null`"
DEBUG="Y"
EUCALYPTUS="/"

usage () {
        echo "$0 [options]"
        echo
        echo "   -help                          this message"
        echo "   -type [nc|walrus|sc]           which stat file to look for"
        echo "   -d <eucalyptus_dir>            where eucalyptus is installed"
        echo
}
# let's parse the command line
while [ $# -gt 0 ]; do
        if [ "$1" = "-h" -o "$1" = "-help" -o "$1" = "?" -o "$1" = "--help" ]; then
                usage
                exit 0
        fi
        if [ "$1" = "-type" ]; then
                if [ -z "$2" ]; then
                        echo "Need the type!"
                        exit 1
                fi
                TYPE="$2"
                shift; shift
                continue
        fi
        if [ "$1" = "-debug" ]; then
                DEBUG="Y"
                shift
                continue
        fi
        if [ "$1" = "-d" ]; then
```

```
                if [ -z "$2" ]; then
                        echo "Need eucalyptus directory!"
                        exit 1
                fi
                EUCALYPTUS="$2"
                shift; shift
                continue
        fi
        usage
        exit 1
done

# some checks
if [ -z "$GMETRIC" ]; then
        echo "Cannot find gmetric: do you have ganglia installed?"
        exit 1
fi
if [ ! -e "$EUCALYPTUS/etc/eucalyptus/eucalyptus.conf" ]; then
        echo "Is Eucalyptus installed in $EUCALYPTUS?"
        exit 1
fi
if [ "$TYPE" = "nc" ]; then
        # let's check we have the stat file
        if [ ! -e ${EUCALYPTUS}${NC_STAT} ]; then
                echo "Cannot find NC stat file!"
                exit 1
        fi

        # number of running VMs
        NUM="`grep ^id: ${EUCALYPTUS}${NC_STAT}|wc -l`"
        # memory available to VMs
        M_AVAIL="`grep ^memory ${EUCALYPTUS}${NC_STAT}|sed
\"s;memory.*MB: [[:digit:]]*/\([[:digit:]]*\)/[[:digit:]]*;\1;\"`"
```

```
        # memory used by VMs
        M_USED="`grep ^memory ${EUCALYPTUS}${NC_STAT}|sed \"s;memory.*MB:
[[:digit:]]*/[[:digit:]]*/\([[:digit:]]*\);\1;\"`"
        # cores available to VMs
        C_AVAIL="`grep ^cores ${EUCALYPTUS}${NC_STAT}|sed \"s;cores.*:
[[:digit:]]*/\([[:digit:]]*\)/[[:digit:]]*;\1;\"`"
        # cores used by VMs
        C_USED="`grep ^cores ${EUCALYPTUS}${NC_STAT}|sed \"s;cores.*:
[[:digit:]]*/[[:digit:]]*/\([[:digit:]]*\);\1;\"`"
        # disk available to VMs
        D_AVAIL="`grep ^disk ${EUCALYPTUS}${NC_STAT}|sed \"s;disk.*GB:
[[:digit:]]*/\([[:digit:]]*\)/[[:digit:]]*;\1;\"`"
        # disk used by VMs
        D_USED="`grep ^disk ${EUCALYPTUS}${NC_STAT}|sed \"s;disk.*GB:
[[:digit:]]*/[[:digit:]]*/\([[:digit:]]*\);\1;\"`"

        [ "$DEBUG" = "Y" ] && echo $GMETRIC -n "Running VMs" -v $NUM -t
int16
        [ "$DEBUG" = "Y" ] && echo $GMETRIC -n "VMs available memory" -v
$M_AVAIL -t int32 -u MB
        [ "$DEBUG" = "Y" ] && echo $GMETRIC -n "VMs used memory" -v
$M_USED -t int32 -u MB
        [ "$DEBUG" = "Y" ] && echo $GMETRIC -n "VMs available cores" -v
$C_AVAIL -t int32
        [ "$DEBUG" = "Y" ] && echo $GMETRIC -n "VMs used cores" -v $C_USED
-t int32
        [ "$DEBUG" = "Y" ] && echo $GMETRIC -n "VMs available disks" -v
$D_AVAIL -t int32 -u GB
        [ "$DEBUG" = "Y" ] && echo $GMETRIC -n "VMs used disks" -v $D_USED
-t int32 -u GB

        $GMETRIC -n "Running VMs" -v $NUM -t int16
        $GMETRIC -n "VMs available memory" -v $M_AVAIL -t int32 -u MB
```

```
        $GMETRIC -n "VMs used memory" -v $M_USED -t int32 -u MB
        $GMETRIC -n "VMs available cores" -v $C_AVAIL -t int32
        $GMETRIC -n "VMs used cores" -v $C_USED -t int32
        $GMETRIC -n "VMs available disks" -v $D_AVAIL -t int32 -u GB
        $GMETRIC -n "VMs used disks" -v $D_USED -t int32 -u GB

elif [ "$TYPE" = "sc" ]; then
        # let's check we have the stat file
        if [ ! -e ${EUCALYPTUS}${SC_STAT} ]; then
                echo "Cannot find SC stat file!"
                exit 1
        fi

        V_USED="`tail -n 1 ${EUCALYPTUS}${SC_STAT}|sed \"s/.*Volumes:
\([[:digit:]]*\).*/\1/\"`"
        S_USED="`tail -n 1 ${EUCALYPTUS}${SC_STAT}|sed \"s/.*Space Used:
\([[:digit:]]*\)/\1/\"`"

        [ "$DEBUG" = "Y" ] && echo $GMETRIC -n "Volumes" -v $V_USED -t
int16
        [ "$DEBUG" = "Y" ] && echo $GMETRIC -n "Volumes disk usage" -v
$S_USED -t int16

        $GMETRIC -n "Volumes" -v $V_USED -t int16
        $GMETRIC -n "Volumes disk usage" -v $S_USED -t int16

elif [ "$TYPE" = "walrus" ]; then
        # let's check we have the stat file
        if [ ! -e ${EUCALYPTUS}${WALRUS_STAT} ]; then
                echo "Cannot find wlarus stat file!"
                exit 1
        fi
```

```
        B_USED="`tail -n 1 ${EUCALYPTUS}${WALRUS_STAT}|sed \"s/.*Buckets:
\([[:digit:]]*\).*/\1/\"`"
        S_USED="`tail -n 1 ${EUCALYPTUS}${WALRUS_STAT}|sed \"s/.*Space
Used: \([[:digit:]]*\)/\1/\"`"

        [ "$DEBUG" = "Y" ] && echo $GMETRIC -n "Buckets" -v $B_USED -t int16
        [ "$DEBUG" = "Y" ] && echo $GMETRIC -n "Buckets disk usage" -v
$S_USED -t int16

        $GMETRIC -n "Buckets" -v $B_USED -t int16
        $GMETRIC -n "Buckets disk usage" -v $S_USED -t int16
else
        echo "Unknown type!"
        exit 1
fi
```

위 스크립트는 노드 컨트롤러 혹은 스토리지 컨트롤러를 모니터링하기 위해서 사용될 수 있으며 crond에 의해서 주기적으로 실행되어야 한다. 노드 컨트롤러를 모니터링하기 위해서는 다음과 같이 실행한다.

```
sh /root/ganglia-bin/ganglia.sh -type nc
```

스토리지 컨트롤러를 모니터링하기 위해서는 다음과 같이 실행한다.

```
sh /root/ganglia-bin/ganglia.sh -type sc
```

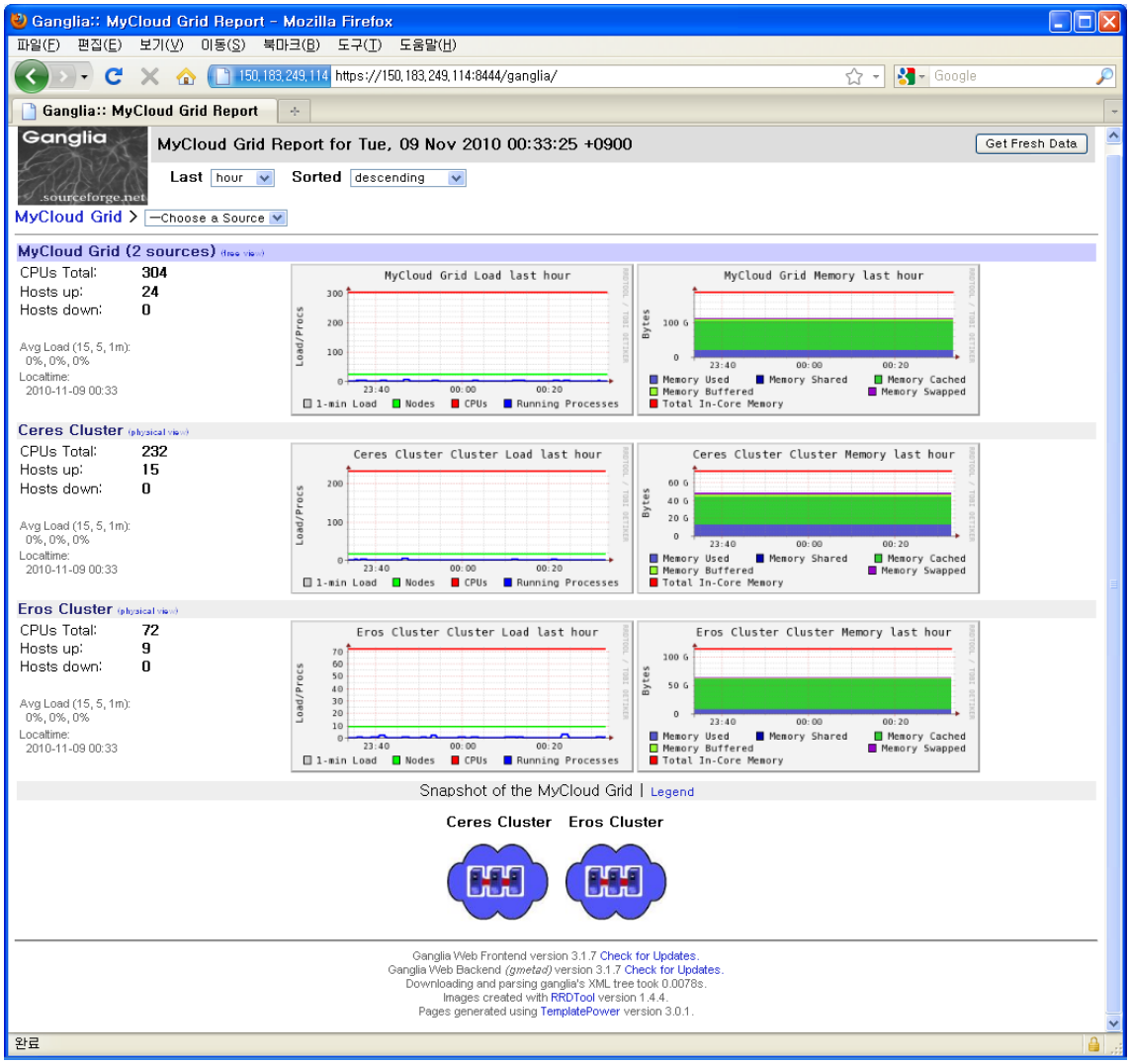

그림 3 사이언스 클라우드 모니터링 시스템 구축

사이언스 클라우드 모니터링 시스템에서 가상화된 컴퓨팅 환경과 스토리지 공간을 모니터링한 결과로 실행 VM의 개수, VM이 이용가능한 CPU 코어 개수, 디스크 크기, 메모리 크기, VM이 사용중인 CPU 코어 개수, 디스크 크기, 메모리 크기 등이 모니터링 가능하게 되었다. 이 외에도 필요한 정보가 있다면 간단한 쉘스크립트와 gmetric을 이용해서 ganglia 정보를 확장하는 것이 용이하다.

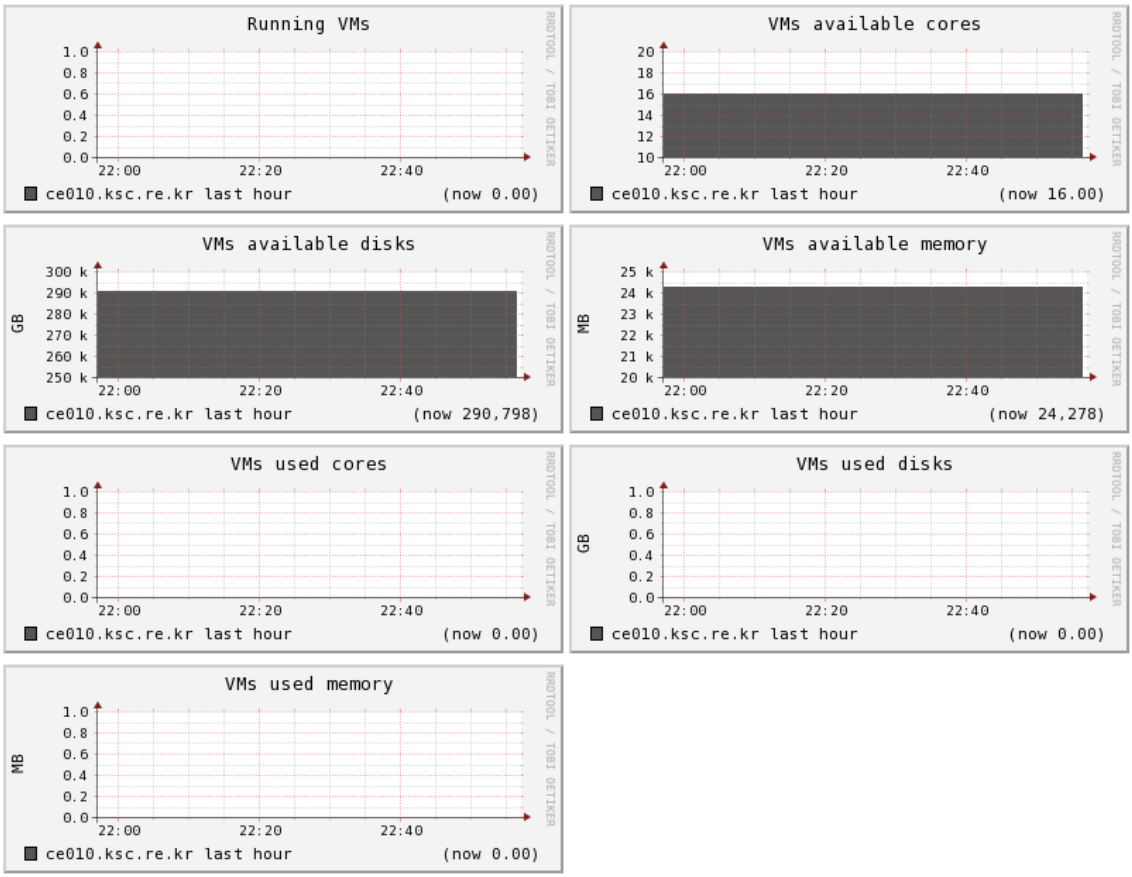

그림 4 가상화된 컴퓨팅 환경 및 스토리지 공간 모니터링

제 8 장 결론

지금까지 Ganglia를 이용하여 이기종 클라우드 환경에서의 모니터링 시스템을 구축하기 위한 방법을 설명하고 gmetric을 이용한 Ganglia 모니터링 요소 확장 기능을 활용하여 사이언스 클라우드 모니터링 시스템을 구축한 사례를 설명하였다.

작년부터 클라우드 컴퓨팅이 IT업계의 최대 화두로 떠오르면서 과학 응용 분야에서도 클라우드 컴퓨팅을 적용하려는 시도들이 점차적으로 생겨나고 있다. 대표적인 상용 클라우드 컴퓨팅 서비스인 아마존 웹서비스의 경우, 이미 몇가지 과학 분야 어플리케이션을 수행하고 이를 기반으로 과학 응용 분야에서 클라우드 컴퓨팅 활용 가능성을 분석한 사례들도 보이고 있으며, 대표적인 오픈 소스 기반 인프라스트럭처 클라우드 소프트웨어 개발 프로젝트인 Eucalyptus, OpenNebula, Nimbus 등에서도 과학 응용 커뮤니티와의 협업을 통해서 다양한 활용 사례를 보고하고 있다. 클라우드 컴퓨팅이 완전히 새로운 형태의 IT 환경을 제공하고 있는 만큼 과학 응용 분야에서도 클라우드 컴퓨팅에 대한 지속적인 관심이 요구된다.