GangaKISTI
설치 및 활용 가이드

(SL4, gLite-UI, Globus, Gridway, Ganga)

2009. 11

차세대연구환경개발실 기술개발팀

한국과학기술정보연구원
Korea Institute of Science and Technology Information
www.kisti.re.kr

머 리 말

본 매뉴얼은 CERN에서 개발된 Ganga와 KISTI에서 개발된 GangaKISTI backend를 활용하여 3개 이기종 인프라(EGEE, PRAGMA, KISTI 슈퍼컴퓨터)를 활용하는 방법을 설명하기 위해서 작성되었습니다. 본 매뉴얼의 작성에 도움을 주신 모든 분들께 감사드립니다.

작 성 자 : 황 순 욱 (KISTI)
김 한 기 (KISTI)

목 차

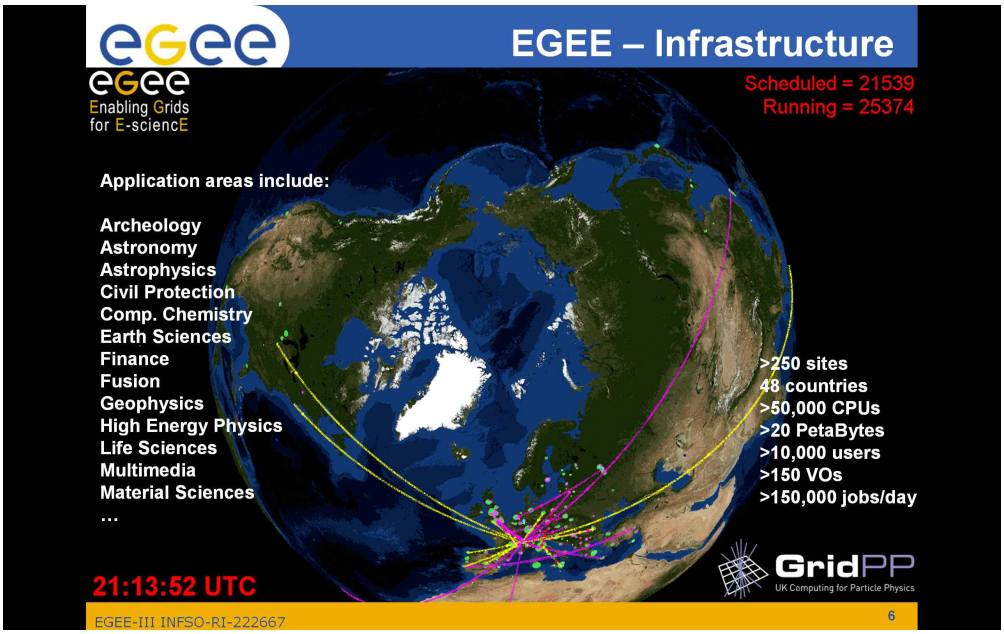
I. 개요 ····· 2
1. EGEE 프로젝트 & gLite ····· 2
2. PRAGMA, Globus & Gridway ····· 4
3. KISTI 슈퍼컴퓨터 Tachyon ····· 7
4. Ganga ····· 8
II. S/W 설치 ····· 10
1. SL4 ····· 10
2. gLite-UI ····· 12
3. Globus ····· 16
4. Gridway ····· 24
5. Scms & Scmsweb ····· 32
6. Ganga ····· 36
III. 3개 이기종 인프라 활용방법 ····· 42
1. EGEE ····· 43
2. PRAGMA ····· 50
3. KISTI 슈퍼컴퓨터 ····· 52

I. 개요

1. EGEE 프로젝트 & gLite

1.1 EGEE 현황 (2009. 5. 1 기준)

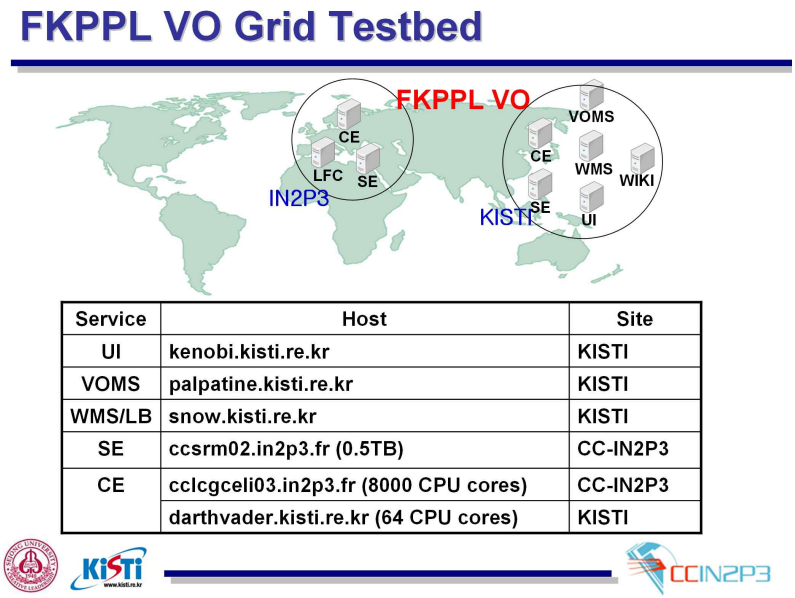
- o 50 개국, 250 사이트
- o 68,000 CPU 사용가능, 24시간 일주일 내내
- o 20 Petabytes (20 million Gigabytes) 저장 공간
- o 평균 작업 로드 150K작업/하루, 최대 188K작업/하루
- o 대용량 작업 전송 초당 1.5 GB 이상
- o 활동 중인 144 이상의 VO (Virtual Organization)



1.2 gLite

gLite는 CERN의 LHC 실험 및 여러 과학 분야에서 사용 중인 그리드 컴퓨팅을 위한 미들웨어 소프트웨어이다. EGEE 프로젝트의 일환으로 12 개의 학계 및 산업계 연구기관의 80명 이상의 사람들의 협력으로 만들어졌으며, 프로덕션 그리드 인프라(production grid infrastructure)를 만들기 위한 완전한 서비스 셋을 제공한다.

현재 KISTI의 FKPPL VO(Virtual Organization)에서는 gLite 3.1.0 버전을 사용하고 있으며, gLite에 대한 보다 자세한 설명은 KISTI에서 발간된 “ALICE COMPUTING MIDDLEWARE 설치 및 활용 가이드"(ISBN:978-89-6211-3617)을 참고하기 바란다.



2. PRAGMA, Globus & Gridway

PRAGMA
PACIFIC RIM APPLICATIONS AND GRID MIDDLEWARE ASSEMBLY

2.1 PRAGMA (Pacific Rim Applications and Grid Middleware Assembly)

PRAGMA 는 태평양 주변 국가의 선도적인 연구 기관들 간에 지속적인 협력과 그리드 기술의 사용을 촉진하기 위해서 2002년에 만들어졌다. 현재 35개 기관이 가입되어 있으며, 2년에 두 번 PRAGMA Workshop을 통해서 모임을 가지고 있다. PRAGMA에서는 응용 프로그램(application)이 핵심(key)이며, 필요한 인프라와 미들웨어는 응용 프로그램의 목표를 달성하는데 초점을 맞춰 구축되고 있다.

진행중인 프로젝트

* Avian Flu Grid
* PRIME
* PRIUS
* GLEON
* Global Cyberbridges

2.2 Globus

Globus는 오픈 소스 그리드 미들웨어로 그리드 컴퓨팅 환경을 구축하는데 사용된다. 현재 Globus Alliance에 의해서 개발되고 있으며 전 세계적으로 여러 기관 및 프로젝트에서 많이 사용되고 있다.

Globus를 미들웨어로 사용중인 프로젝트

- Open Science Grid
- TERAGRID
- PRAGMA

2.3 Gridway

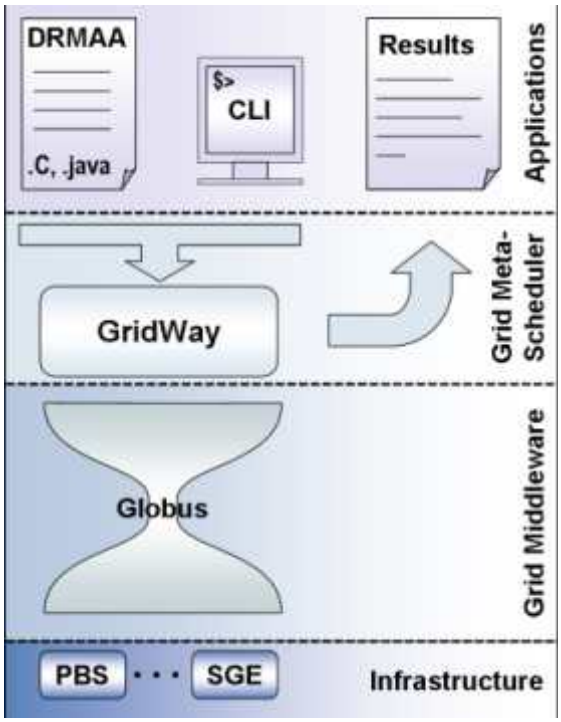
Gridway는 Globus Toolkit의 메타 스케줄링 컴포넌트로 GRAM(Grid Resource Allocation and Management), MDS(Monitoring and Discovery Service), GridFTP의 Globus 서비스 위에서 스케줄러 가상화 계층(scheduler virtualization layer)을 제공한다.

오픈 소스(Open source) 프로젝트로 최종 사용자(end user)나 응용 프로그램 개발자에게 로컬 DRM(Distributed Resource Management) 시스템에서 제공하는 기능들, 즉 작업 제출, 모니터링, 동기화, 작업 제어하기 등의 작업 스케줄링 기능을 제공한다. Gridway는 작업 스케줄링을 위해 CLI(command line interface)와 OGF 표준 방식인

DRMAA(Distributed Resource Management Application API)를 모두 제공한다.

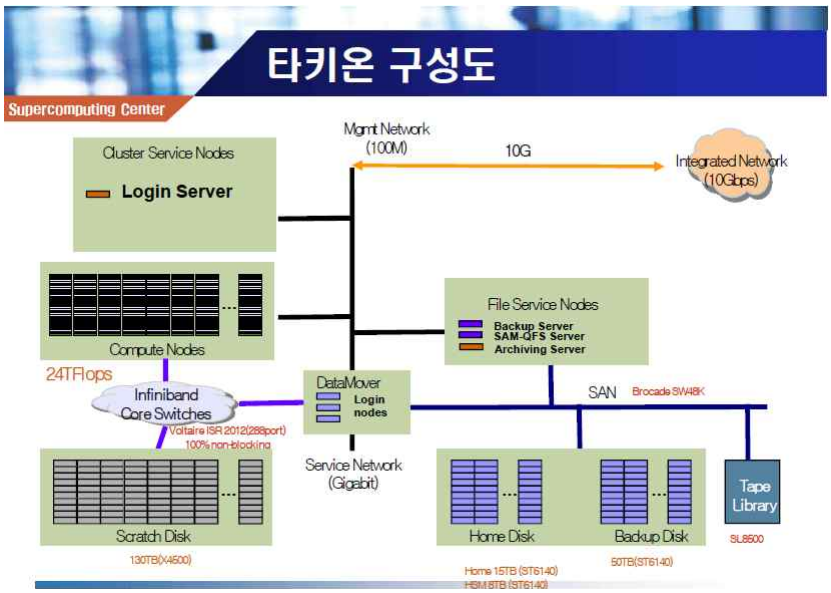
Gridway는 서로 다른 그리드 서비스를 연동하기 위해서 미들웨어 접속 드라이버(Middleware Access Drivers)를 사용한다. 기본적으로 다음의 MADs가 제공된다.

- pre-WS GRAM과 WS GRAM 서비스와 연동하는 (interface) 실행 매니저(Execution Manager)
- MDS2(MDS and GLUE schema)와 MDS4 서비스와 연동하는 정보 매니저(Information Manager)
- GridFTP 서비스와 연동하는 전송 매니저



3. KISTI 슈퍼컴퓨터 Tachyon

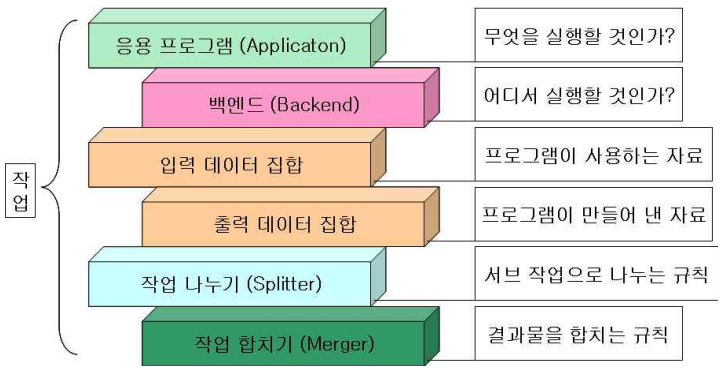
KISTI 슈퍼컴퓨터 4호기 타키온은 CentOS 기반의 클러스터로 SGE(Sun Grid Engine)을 스케줄러로 사용하고 있으며, Lustre 공유 파일 시스템을 사용하고 있다.



4. Ganga

Ganga는 파이썬으로 구현된 작업 정의 및 관리 도구로 사용하기 편한 작업 환경을 제공한다. Ganga는 GAUIDI/ATHENA and Grid Alliance의 약어로, GAUDI와 ATHENA는 cern의 4대 실험에 속하는 ATLAS, LHCb 실험에서 공통으로 사용되는 프레임워크(framework) 이다. Ganga라는 이름에서 알 수 있듯이 초기에는 ATLAS 실험과 LHCb 실험에 참여하는 물리학자들에게 사용하기 쉬운 그리드 사용자 인터페이스를 제공해 주고자 하는 목적으로 개발되었으나, 현재는 고에너지 물리 분야뿐만 아니라 Geant4 regression 테스트나 웹 검색 이미지 분류 등 범용의 목적으로 사용되고 있다. Ganga는 실험 결과의 분산 해석 시스템(distributed-analysis system)을 주요 기능으로 제공하고 있다.

Ganga에서 하나의 작업은 아래 그림에서와 같이 여러 개의 빌딩 블록으로 이루어져 있다. 모든 작업에는 실행 하고자 하는 소프트웨어(응용 프로그램)와 작업을 수행하고자 하는 시스템(백엔드)이 정확히 명시되어야 한다.



Ganga에서는 현재 LHC 실험에서 사용 중인 여러 응용 프로그램을 지원하고 있다. 대표적인 프로그램으로 ROOT, Gauss, Boole, Brunel, DaVinci 등이 있다. 또한 로컬(Local) 백엔드를 기본으로 PBS, LSF, Condor, LCG, DIRAC, gLite, PANDA, ARC 등을 백엔드로 사용가능하다.

KISTI에서는 Globus 미들웨어를 사용하는 PRAGMA 자원을 Ganga를 사용해서 활용하기 위해서 Gridway 백엔드 플러그인을 개발하였고, 또한 gLite 미들웨어를 사용하는 EGEE 자원과 Globus 미들웨어를 사용하는 PRAGMA 자원을 각 자원의 로드(CPU 사용량)에 따라 선택적으로 사용할 수 있는 Intergrid 백엔드 플러그인 개발하여, GangaKISTI 라는 이름으로 Ganga 5.3.2 배포판에 포함시켜 배포하였다.

http://ganga.web.cern.ch/ganga/release/5.3.2/release/ReleaseNotes-5.3.2

GangaKISTI 모듈에는 Gridway 백엔드와 Intergrid 백엔드의 활용성을 보여주기 위한 application 플러그인으로 Autodock applicaion이 함께 포함되어 있다.

II. S/W 설치

1. SL4 (Scientific Linux 4) 설치

1.1 SL4의 최신버전 SL48

ftp://ftp.scientificlinux.org/linux/scientific/48/iso/i386/dvd/

Scientific Linux which in turn is rebuilt from the freely available RHEL4(Red Hat Enterprise Linux 4)

자세한 설명은 아래 설치 가이드를 참고한다.

Installation guide for Scientific Linux v4.x

https://www.scientificlinux.org/distributions/4x/installing/install-guide-full.html

※ 최신 Linux의 DVD 버전이 제공되지 않을 때

CD ISO 이미지를 사용해서 DVD ISO 이미지 만들기

http://www.certifried.com/?q=node/5

만들어진 ISO 이미지를 DVD로 굽기

http://www.imgburn.com/index.php

※ 설치시 유의사항

Installation Type은 Custom 으로

Security Enhanced Linux는 Disabled 시킨다.

Package Group Selection에서 다음의 패키지를 포함해서 설치한다.

X Window System
GNOME Desktop Environment
OpenAFS Client
YUM
Editors
Engineering and Scientific
Graphical Internet
Text-based Internet
Sound and Video
Graphics
Server Configuration Tools
Mail Server
Development Tools
Administration Tools
System Tools

1.2 Manual post-install configuration adjustment

· 실행중인 서비스 확인하기

/sbin/chkconfig --list

· 서비스 동작 중지하기

/sbin/chkconfig servicename off 나 /sbin/service servicename stop

· 서비스 동작 다시 시작하기

/sbin/chkconfig servicename on

· 방화벽(iptables) 설정

```
service iptables stop
vi /etc/sysconfig/iptables
필요한 포트 open
service iptables start
```

· ntp 설정

```
echo "restrict 210.98.16.100 mask 255.255.255.255 nomodify notrap noquery" >> /etc/ntp.conf
echo "server time.nuri.net" >> /etc/ntp.conf
echo "time.nuri.net" >> /etc/ntp/step-tickers
ntpdate time.nuri.net
ntpq -p
service ntpd start
chkconfig ntpd on
```

ntp 서버리스트는 정보는 다음 웹사이트에서 확인 가능, pool.ntp.org: the internet cluster of ntp servers

방화벽 문제로 ntp 서버를 이용할 수 없을 때에는 rdate 명령으로 시간을 맞춘다.

$rdate time.bora.net

2. gLite-UI (User-Interface)

2.1 CERN 소프트웨어 설치를 위한 yum repository 파일 작성

cd /etc/yum.repos.d/

vi jpackage5.0.repo

```
[jpackage17-generic]
name=JPackage 1.7, generic
baseurl=http://mirrors.dotsrc.org/jpackage/1.7/generic/free/
enabled=1
protect=1
gpgkey=http://www.jpackage.org/jpackage.asc
gpgcheck=1

[jpackage17-generic-nonfree]
name=JPackage 1.7, generic non-free
baseurl=http://mirrors.dotsrc.org/jpackage/1.7/generic/non-free/
enabled=1
protect=1
gpgkey=http://www.jpackage.org/jpackage.asc
gpgcheck=1

[jpackage5-generic]
name=JPackage 5, generic
baseurl=http://mirrors.dotsrc.org/jpackage/5.0/generic/free/
enabled=1
protect=1
gpgkey=http://www.jpackage.org/jpackage.asc
gpgcheck=1

[jpackage5-generic-nonfree]
name=JPackage 5, generic non-free
baseurl=http://mirrors.dotsrc.org/jpackage/5.0/generic/non-free/
enabled=1
protect=1
gpgkey=http://www.jpackage.org/jpackage.asc
gpgcheck=1
```

rpm --import http://www.jpackage.org/jpackage.asc

vi dag.repo

```
[main]
[dag]
name=DAG (http://dag.wieers.com) additional RPMS repository
baseurl=http://linuxsoft.cern.ch/dag/redhat/el4/en/$basearch/dag
gpgkey=http://linuxsoft.cern.ch/cern/slc4X/$basearch/docs/RPM-GPG-KEY-dag
gpgcheck=1
enabled=1
```

vi cern.repo

```
[sl-base]
baseurl=http://linuxsoft.cern.ch/scientific/4x/i386/SL/RPMS
enabled=1
protect=1

[slc-base]
baseurl=http://linuxsoft.cern.ch/cern/slc4X/i386/yum/os
enabled=1
protect=0

[slc-update]
baseurl=http://linuxsoft.cern.ch/cern/slc4X/i386/yum/updates
enabled=1
protect=0
```

vi lcg-ca.repo

```
#
# Scientific Linux CERN 4 (SLC4) extra packages
# (not included in base repository)
#
[CA]
name=CAs
baseurl=http://linuxsoft.cern.ch/LCG-CAs/current
```

vi glite-UI.repo

```
# This is the official YUM repository string for the glite 3.1 UI
# Fetched from: http://grid-deployment.web.cern.ch/grid-deployment/yaim/repos/glite-UI.repo
# Place it to /etc/yum.repos.d/ and run 'yum update'

[glite-UI]
name=gLite 3.1 User Interface
baseurl=http://linuxsoft.cern.ch/EGEE/gLite/R3.1/glite-UI/sl4/$basearch/
enabled=1
```

repository 파일을 모두 만든 후, yum update를 실행한다.

yum -y update

※ yum update 시 문제가 발생했을 때 해결방법

(1) Error: Missing Dependency: openssh = 4.1p1-3.hpn_cern_test7 is needed by package openssh-askpass-gnome

충돌이 발생한 패키지를 제외하고 update 실행

yum --exclude=openssh* update

(2) Transaction Check Error: file /usr/lib/perl5/vendor_perl/5.8.5/Compress/Zlib.pm conflicts between attempted installs of perl-IO-Compress-2.021-1.el4.rf and perl-Compress-Zlib-2.015-1.el4.rf

file /usr/share/man/man3/Compress::Zlib.3pm.gz conflicts between attempted installs of perl-IO-Compress-2.021-1.el4.rf and perl-Compress-Zlib-2.015-1.el4.rf

출돌이 발생한 파일을 포함한 rpm 패키지 확인/제거 후, update 실행

rpm -qif /usr/share/man/man3/Compress::Zlib.3pm.gz

yum remove perl-Compress-Zlib

yum --exclude=openssh* update

2.2 lcg-CA, glite-UI 설치

yum -y install lcg-CA

yum -y install glite-UI

※ glite-UI 설치 시 java 패키지 충돌 해결방법

Error: Missing Dependency: jdk = 2000:1.6.0_16-fcs is needed by package java-1.6.0-sun-compat

일단 java-1.6.0-sun-compat 패키지를 제거한다.

yum -y remove java-1.6.0-sun-compat

패키지 정보가 수정된 java rpm을 내려 받아서 localinstall 방식으로 설치한다.

wget http://yoda.kisti.re.kr/sw/java-1.5.0-sun-1.5.0.15-1jpp.i586.rpm

wget http://yoda.kisti.re.kr/sw/java-1.5.0-sun-devel-1.5.0.15-1jpp.i586.rpm

yum -y localinstall java-1.5.0-sun-1.5.0.15-1jpp.i586.rpm

yum -y localinstall java-1.5.0-sun-devel-1.5.0.15-1jpp.i586.rpm

yum -y install glite-UI

2.3 yaim site-info.conf 설정

mkdir -p /opt/glite/yaim/etc

cp /opt/glite/yaim/examples/groups.conf /opt/glite/yaim/etc/

cp /opt/glite/yaim/examples/users.conf /opt/glite/yaim/etc/

cp /opt/glite/yaim/examples/siteinfo/site-info.def /opt/glite/yaim/etc/

기존에 사용 중인 site-info.def를 kenobi.kisti.re.kr에서 복사해 온다.

scp hgkim@kenobi.kisti.re.kr:/opt/glite/yaim/etc/site-info.def /opt/glite/yaim/etc/site-info.def
vi /opt/glite/yaim/etc/site-info.def

/opt/glite/yaim/etc/site-info.def
/opt/glite/yaim/bin/yaim -c -s /opt/glite/yaim/etc/site-info.def -n UI

2.4 설치확인

$glite-version
3.1.0

3. Globus

3.1 Globus 설치

설치할 globus는 4.0.3 버전으로 globus alliance 홈페이지에서 내려 받으면 된다.

http://www.globus.org/toolkit/downloads/4.0.3/#source
Source Download
Download page for gt4.0.3-all-source-installer.tar.gz

3.1.1 Prerequisite

JDK 1.5 이상, Apache Ant

Ant 설치 방법

① 수동 설치

```
su -
cd /usr/local/
wget http://mirror.apache.or.kr/ant/binaries/apache-ant-1.7.1-bin.tar.gz
tar xzvf apache-ant-1.7.1-bin.tar.gz
```

② yum을 이용한 방법

yum install ant ant-apache-regexp ant-trax ant-nodeps

참고페이지

HOWTO Install the GridWay Metascheduler

https://wiki.gridx1.ca/twiki/bin/view/Main/InstallingGridWay

3.1.2 globus 사용자 계정 만들기

globus 설치를 위한 사용자 계정(globus)을 만들고, 이 계정(globus)을 사용해서 나머지 설치를 진행한다.

```
su -
adduser globus
passwd globus
mkdir /usr/local/globus-4.0.3
chown globus:globus /usr/local/globus-4.0.3
```

3.1.3 globus compile & install

globus 컴파일에 필요한 환경변수 설정

```
su - globus
vi .bashrc
export JAVA_HOME=/usr/java/jdk1.5.0_15/
export PATH=${JAVA_HOME}/bin:$PATH
export ANT_HOME=/usr/local/apache-ant-1.7.1
export PATH=${ANT_HOME}/bin:$PATH
export GLOBUS_LOCATION=/usr/local/globus-4.0.3
source .bashrc
```

JAVA_HOME 환경변수는 시스템에 설치된 JDK의 위치를 확인해서 알맞게 설정한다.

```
rpm -qil java-1.5.0-sun-devel | grep bin/java
/usr/lib/jvm/java-1.5.0-sun-1.5.0.15/bin/java
/usr/lib/jvm/java-1.5.0-sun-1.5.0.15/bin/javac

export JAVA_HOME=/usr/lib/jvm/java-1.5.0-sun-1.5.0.15
```

globus 소스 파일을 내려 받아서, 컴파일 및 설치 진행

```
wget http://www-unix.globus.org/toolkit/survey/index.php?download=gt4.0.3-all-source-installer.tar.gz
tar zxvf gt4.0.3-all-source-installer.tar.gz
cd gt4.0.3-all-source-installer
./configure --prefix=$GLOBUS_LOCATION
make
make install
```

3.1.4 Sharded Library 설정

gLite middleware가 globus를 기반으로 만들어졌기 때문에 gLite UI가 설치된 시스템에 globus를 설치하면, 두 middleware의 Shared Library간의 충돌이 발생할 수 있다. 이를 해결하기 위해서 다음과 같이 /etc/ld.so.conf 파일을 수정해 준다.

```
su -
vi /etc/ld.so.conf
```

globus의 shared library path를 제일 위에 추가한다.

```
include ld.so.conf.d/*.conf
/usr/local/globus-4.0.3/lib
/opt/lcg/lib
/opt/glite/lib
/opt/globus/lib
/usr/local/lib
/usr/X11R6/lib
```

ldconfig 명령을 실행해서 수정된 conf 파일을 시스템에 반영한다.

3.1.5 globus 설치 후 동작확인

● globus 버전 확인

```
$ which globus-version
/usr/local/globus-4.0.3/bin/globus-version
$ globus-version
4.0.3
```

● proxy certificate 만들기

proxy certificate를 만들기 위해서는 .globus 폴더 밑에 본인의 certificate를 가지고 있어야 한다.

```
$ ll .globus
-rw-r--r--  1 hgkim hgkim 1810 Sep  8 09:51 usercert.pem
-r--------  1 hgkim hgkim 1995 Sep  8 09:51 userkey.pem
```

인증서(certificate) 발급에 관련된 사항은 아래 홈페이지를 참고한다.

※ 인증서 발급과정

http://esgtech.springnote.com/pages/2779308

```
$ grid-proxy-destroy
$ grid-proxy-init
Your identity: /C=KR/O=KISTI/O=GRID/O=KISTI/CN=55879660 HanGi kim
Enter GRID pass phrase for this identity:
Creating proxy ..................................... Done
Your proxy is valid until: Mon Nov 16 23:43:11 2009
```

```
$ grid-proxy-info -all
subject  : /C=KR/O=KISTI/O=GRID/O=KISTI/CN=55879660 HanGi kim/CN=14455635
issuer   : /C=KR/O=KISTI/O=GRID/O=KISTI/CN=55879660 HanGi kim
identity : /C=KR/O=KISTI/O=GRID/O=KISTI/CN=55879660 HanGi kim
type     : Proxy draft (pre-RFC) compliant impersonation proxy
strength : 512 bits
path     : /tmp/x509up_u500
timeleft : 11:57:55
```

● 인증(Authorization) 테스트

```
$ globusrun -a -r kyle.kisti.re.kr

GRAM Authentication test successful
```

● 작업 제출(Submission) 테스트

```
$ globus-job-run kyle.kisti.re.kr /bin/uname -a
Linux kyle.kisti.re.kr 2.6.18-92.el5 #1 SMP Tue Jun 10 18:49:47 EDT 2008 i686
i686 i386 GNU/Linux
```

● 파일 전송 테스트

```
$ globus-url-copy gsiftp://kyle/etc/group file:///home/hgkim/group
$ globus-url-copy file:///home/hgkim/group gsiftp://kyle/tmp/group
```

※인증이나 globus-url-copy 명령 사용시 문제가 발생할 경우, -vb -dbg 옵션을 함께 사용해서 문제점이 무엇인지 파악한다.

globus-url-copy **-vb -dbg** gsiftp://kyle/etc/group file:///home/hgkim/group

· Globus WS installation 동작 확인

● 작업 제출(Submission) 테스트

```
$ globusrun-ws -F kyle -submit -c /bin/true
Submitting job...Done.
Job ID: uuid:6f54d918-d26f-11de-bf62-0016e66500c5
Termination time: 11/17/2009 05:18 GMT
Current job state: Active
Current job state: CleanUp
Current job state: Done
Destroying job...Done.
$ echo $?
0
$ globusrun-ws -F kyle -submit -c /bin/false
Submitting job...Done.
Job ID: uuid:753020ae-d26f-11de-a57b-0016e66500c5
Termination time: 11/17/2009 05:18 GMT
Current job state: Active
Current job state: CleanUp
Current job state: Done
Destroying job...Done.
$ echo $?
1
$ globusrun-ws -submit -F kyle -factory-type Fork -s -job-command
/bin/hostname
Delegating user credentials...Done.
Submitting job...Done.
Job ID: uuid:a3008690-d26f-11de-98a8-0016e66500c5
Termination time: 11/17/2009 05:19 GMT
Current job state: Active
Current job state: CleanUp-Hold
kyle.kisti.re.kr
Current job state: CleanUp
Current job state: Failed
Destroying job...Done.
Cleaning up any delegated credentials...Done.
globusrun-ws: Job failed: Staging error for RSL element fileCleanUp.
```

```
$ globusrun-ws -submit -F kyle -factory-type SGE -s -job-command /bin/hostname
Delegating user credentials...Done.
Submitting job...Done.
Job ID: uuid:b6dff22c-d26f-11de-a1bb-0016e66500c5
Termination time: 11/17/2009 05:20 GMT
Current job state: Pending
Current job state: Active
kyle02.kisti.re.kr
Current job state: CleanUp-Hold
Current job state: CleanUp
Current job state: Failed
Destroying job...Done.
Cleaning up any delegated credentials...Done.
globusrun-ws: Job failed: Staging error for RSL element fileCleanUp.
```

● Information retrieval 테스트

```
$ wsrf-query -s https://kyle.kisti.re.kr:8443/wsrf/services/DefaultIndexService
<ns0:IndexRP xmlns:glue="http://mds.globus.org/glue/ce/1.1"
xmlns:ns0="http://mds.globus.org/index"
xmlns:ns1="http://docs.oasis-open.org/wsrf/20
04/06/wsrf-WS-ServiceGroup-1.2-draft-01.xsd"
xmlns:ns10="http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceLifetime-1.2
-draft-01.xsd" xmlns
:ns2="http://schemas.xmlsoap.org/ws/2004/03/addressing"
xmlns:ns3="http://www.globus.org/namespaces/2004/10/gram/job"
xmlns:ns4="http://mds.globu
s.org/inmemoryservicegroup" xmlns:ns5="http://mds.globus.org/aggregator/types"
xmlns:ns6="http://mds.globus.org/aggregator" xmlns:ns7="http://mds
.globus.org/metrics/2004/09" xmlns:ns8="http://mds.globus.org/metadata/2005/02"
xmlns:ns9="http://docs.oasis-open.org/wsn/2004/06/wsn-WS-BaseNoti
fication-1.2-draft-01.xsd" xmlns:rft="http://www.globus.org/namespaces/2004/10/rft"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:n
s11="http://mds.globus.org/aggregator/types">
 <ns1:Entry>
... 중략 ...
   <ns11:AggregatorData>
    <ns1:GLUECE xmlns:ns1="http://mds.globus.org/glue/ce/1.1">
     <ns1:ComputingElement ns1:Name="default" ns1:UniqueID="default">
      <ns1:Info ns1:GRAMVersion="4.0.3" ns1:LRMSType="Fork"
ns1:LRMSVersion="1.0" ns1:TotalCPUs="4"/>
      <ns1:State ns1:EstimatedResponseTime="0" ns1:FreeCPUs="0"
ns1:RunningJobs="0" ns1:Status="enabled" ns1:TotalJobs="0" ns1:WaitingJobs="0" ns
1:WorstResponseTime="0"/>
      <ns1:Policy ns1:MaxCPUTime="-1" ns1:MaxRunningJobs="-1"
ns1:MaxTotalJobs="-1" ns1:MaxWallClockTime="-1" ns1:Priority="0"/>
     </ns1:ComputingElement>
    </ns1:GLUECE>
   </ns11:AggregatorData>
... 중략 ...
 <ns1:TerminationTime xsi:nil="true"
xmlns:ns1="http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceLifetime-1.2-
draft-01.xsd"/>
 <ns1:CurrentTime
xmlns:ns1="http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceLifetime-1.2-
draft-01.xsd">2009-11-16T05:21:48.145Z</ns1:Cur
rentTime>
```

4. Gridway 설치

4.1 Globus 설치 확인

3.1.5 globus 설치 후 동작확인이 모두 제대로 동작하는 확인한다.

4.2 Gridway 프로그램 내려받기

Gridway 프로그램을 아래 사이트에서 내려 받는다.

http://www.gridway.org/doku.php?id=software

http://dev.gridway.org/attachments/download/38/gw-5.4.0.tar.gz

※ 현재 Gridway는 5.4.0 버전까지 개발되었으나, 여기서는 호환성 문제 때문에 5.2.3 버전을 사용하도록 한다.

http://yoda.kisti.re.kr/~hgkim/gw-5.2.3.tar.gz

4.3 Gridway compile & install

Gridway의 설치는 단일 사용자 모드와 다중 사용자 모드, 두 가지 모드로 설치가능하다.

① 단일 사용자 모드(Single-User Mode)

Gridway가 설치 될 디렉토리를 만든다.

```
su -
mkdir -p /usr/local/gw
chown globus:globus /usr/local/gw
```

```
su - globus
cd /home/globus
source /usr/local/globus-4.0.3/etc/globus-devel-env.sh
export GW_LOCATION=/usr/local/gw
export LD_LIBRARY_PATH=/usr/local/globus-4.0.3/lib
wget http://yoda.kisti.re.kr/~hgkim/gw-5.2.3.tar.gz
tar xzvf gw-5.2.3.tar.gz
cd gw-5.2.3
./configure --prefix=$GW_LOCATION --with-docs --enable-prews --enable-debug
make
make install
```

컴파일 및 설치가 완료되면 gridway 실행에 필요한 권한을 설정해 준다. (visudo - edit the sudoers file)

```
su -
#visudo (혹은 vim /etc/sudoers)
```

/etc/sudoers 파일에 다음 내용을 추가한다.

```
#Defaults requiretty
Defaults env_keep = "COLORS DISPLAY HOSTNAME HISTSIZE INPUTRC KDEDIR ₩
GLOBUS_LOCATION GW_LOCATION ₩
# GridWay entries
globus ALL=(ALL) NOPASSWD: /usr/local/gw/bin/gw_em_mad_prews *
globus ALL=(ALL) NOPASSWD: /usr/local/gw/bin/gw_em_mad_ws *
globus ALL=(ALL) NOPASSWD: /usr/local/gw/bin/gw_tm_mad_ftp *
```

② 다중 사용자 모드(Multiple-User Mode)

Gridway가 설치 될 디렉토리를 만들고, Gridway 실행 권한을 가진 gwusers 그룹을 만든다.

```
su -
groupadd gwusers
mkdir -p /usr/local/gw
chown globus:gwusers /usr/local/gw
su - globus
cd /home/globus
source /usr/local/globus-4.0.3/etc/globus-devel-env.sh
export GW_LOCATION=/usr/local/gw
export LD_LIBRARY_PATH=/usr/local/globus-4.0.3/lib
wget http://yoda.kisti.re.kr/~hgkim/gw-5.2.3.tar.gz
tar xzvf gw-5.2.3.tar.gz
cd gw-5.2.3
./configure --prefix=$GW_LOCATION --with-docs --enable-prews --enable-debug
make
make install
```

컴파일 및 설치가 완료되면 gridway 실행에 필요한 권한을 설정해 준다. (visudo - edit the sudoers file)

```
su -
#visudo (혹은 vim /etc/sudoers)
```

/etc/sudoers 파일에 다음 내용을 추가한다.

```
#Defaults requiretty
Defaults env_keep = "COLORS DISPLAY HOSTNAME HISTSIZE INPUTRC KDEDIR ₩
GLOBUS_LOCATION GW_LOCATION ₩
# GridWay entries
globus ALL=(%gwusers)      NOPASSWD: /home/local/gw/bin/gw_em_mad_prews *
globus ALL=(%gwusers)      NOPASSWD: /home/local/gw/bin/gw_em_mad_ws *
globus ALL=(%gwusers)      NOPASSWD: /home/local/gw/bin/gw_tm_mad_ftp
```

※ 다중 사용자 모드 사용시 사용자는 다음의 사항을 주의한다.

· 사용자 추가시 gwusers 그룹으로 지정

adduser gtest -g gwusers

· 사용자 디렉토리를 읽을 수 있도록 permission을 변경해 준다.

chmod 750 /hom/gtest

drwxr-x--- 4 gtest gwusers 4096 Nov 12 13:31 gtest

· umask 는 002 로 지정되어 있어야 한다.

umask 0002

umask 가 002 이어야, group permission이 다음과 같이 된다. (일반적인 umask는 022)

drwxrwxr-x

4.4 Gridway 환경 설정(configuration)

Gridway 데몬을 실행하기 전에 다음과 같이 환경 설정을 해 준다.

vi $GW_LOCATION/etc/gwd.conf

```
# Example MAD Configuration for pre-WS testbeds
#IM_MAD = mds2:gw_im_mad_static:-l etc/im_examples/host.list:gridftp:prews
#EM_MAD = prews:gw_em_mad_prews::rsl
#TM_MAD = gridftp:gw_tm_mad_ftp:

# Example MAD Configuration for WS testbeds
IM_MAD = mds4:gw_im_mad_mds4:-s kyle.kisti.re.kr:gridftp:ws
EM_MAD = ws:gw_em_mad_ws::rsl2
TM_MAD = gridftp:gw_tm_mad_ftp:
```

필요에 따라 pre-WS testbeds나 WS testbeds를 선택적으로 사용한다.

4.5 Gridway 데몬(Daemon) 실행 및 중지

ps 명령으로 Gridway 데몬(gwd)이 실행 중인지 확인하고, 데몬이 실행 중이지 않으면 데몬을 실행한다.

```
$ ps -ef | grep gwd
globus    1959  1835  0 16:36 pts/1    00:00:00 grep gwd
globus   28575     1  0 Nov12 ?       00:01:08 gwd
$ gwd
$ gwd -v
Copyright 2002-2007 GridWay Team, Distributed Systems Architecture
Group, Universidad Complutense de Madrid

GridWay 5.2.3 is distributed and licensed for use under the terms of the
Apache License, Version 2.0 (http://www.apache.org/licenses/LICENSE-2.0).
```

데몬(gwd)의 실행을 중지하고자 할 때에는 pkill 명령으로 데몬 프로세스를 kill 시킨다.

```
$ pkill gwd
```

4.6 Gridway 실행 예제

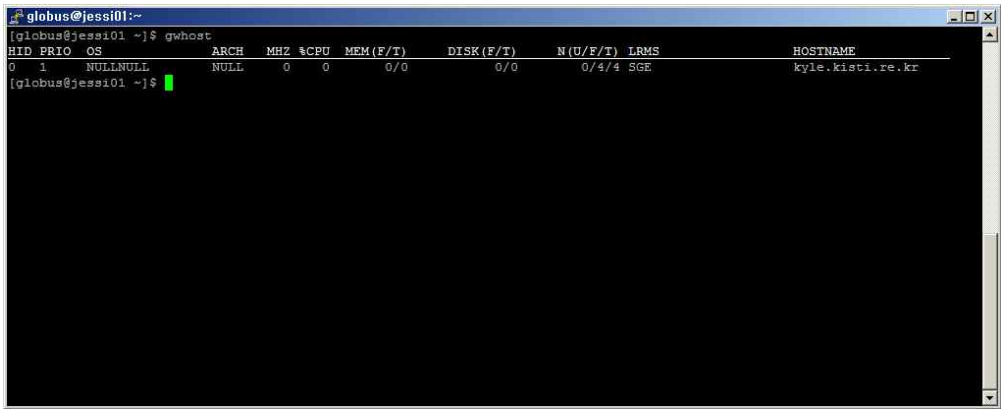
이번에는 Gridway 작업 예제 파일을 가지고 실제로 작업 제출하는 과정을 알아본다.

example.jt

```
EXECUTABLE=/bin/hostname
STDOUT_FILE=stdout
```

먼저 gwhost 명령을 사용해서, 사용 가능한 리소스(resource)가 얼마나 있는지 확인한다.

$gwhost

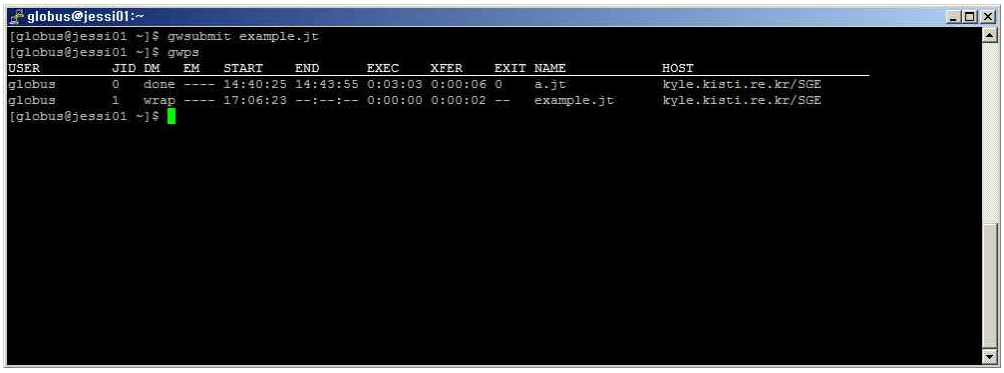


현재 kyle.kisti.re.kr 시스템에서 4개의 CPU가 사용가능하다는 것을 알 수 있다.

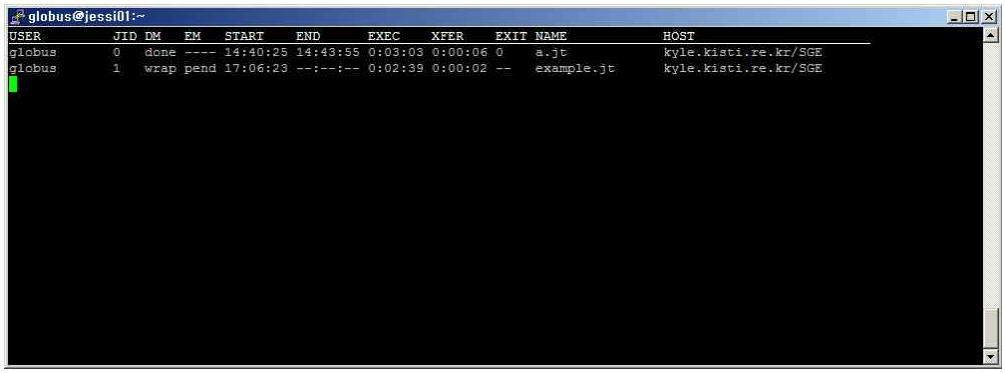
gwsubmit 명령을 사용해서 작업을 제출한다.

$gwsubmit [-t] example.jt

제출된 작업의 상태(status)는 gwps 명령을 사용해서 확인할 수 있다.

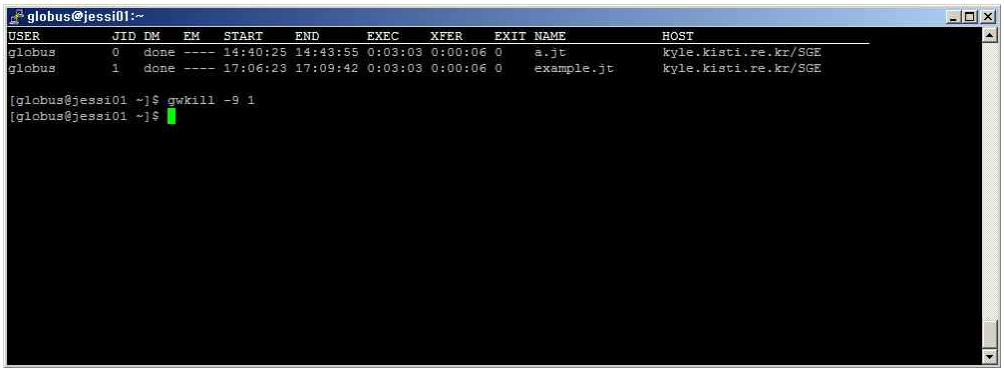


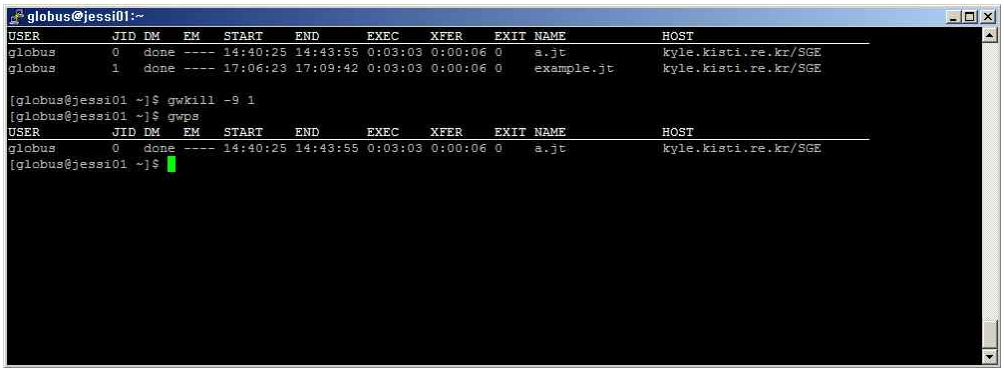
이 때, (-c 시간(초)) 옵션을 사용하면 지정된 시간 간격으로 계속적으로 작업 상태를 확인할 수 있다. (top 명령과 비슷함)



실행 중인 작업은 kill 시키고자 할 때에는 gwkill 명령을 사용한다.

$gwkill -9 job_id





그밖에 Gridway 설치에 관련된 좀 더 자세한 사항이나 문제점이 발생했을 경우에는 아래 문서를 참고하도록 한다.

http://www.gridway.org/doku.php?id=documentation:release_5.4:ig

http://esgtech.springnote.com/pages/2779704

5. Scms & Scmsweb

Gridway를 사용해서 PRAGMA 자원을 연동하기 위해서 Thai National Grid Center에서 개발한 scmsweb-gridway MAD(Middleware Access Drivers)를 사용한다. (현재는 개발이 중지된 상태이다.)

5.1 RPM 설치

scmsweb-gridway를 설치하기 위해서는 scmsweb과 scms 패키지가 필요하다.

현재 프로그램의 개발이 중지된 상태이므로, yoda 웹 서버에 저장된 rpm을 사용하도록 한다.

```
yum install httpd
wget http://yoda.kisti.re.kr/~hgkim/scms/PyChart-1.38-1.noarch.rpm
wget http://yoda.kisti.re.kr/~hgkim/scms/rrdtool-1.0.38-1.i386.rpm
wget http://yoda.kisti.re.kr/~hgkim/scms/PyRRDtool-0.1-5.i386.rpm
wget http://yoda.kisti.re.kr/~hgkim/scms/scebase-1.7.1-1.i386.rpm
wget http://yoda.kisti.re.kr/~hgkim/scms/scebase-devel-1.7.1-1.i386.rpm
wget http://yoda.kisti.re.kr/~hgkim/scms/scms-2.4.0-6.i386.rpm
wget http://yoda.kisti.re.kr/~hgkim/scms/scmsweb-3.2.2-4.noarch.rpm
wget http://yoda.kisti.re.kr/~hgkim/scms/scmsweb-gridway-3.2.2-4.noarch.rpm
rpm -Uvh PyChart-1.38-1.noarch.rpm
rpm -Uvh rrdtool-1.0.38-1.i386.rpm
rpm -Uvh PyRRDtool-0.1-5.i386.rpm
rpm -Uvh scebase-1.7.1-1.i386.rpm
rpm -Uvh scebase-devel-1.7.1-1.i386.rpm
rpm -Uvh scms-2.4.0-6.i386.rpm
rpm -Uvh --nodeps scmsweb-3.2.2-4.noarch.rpm
rpm -Uvh scmsweb-gridway-3.2.2-4.noarch.rpm
```

※ 주의: scmsweb-3.2.2-4.noarch.rpm 은 python 2.4 이상의 버전을 필요로 하나, SL4에는 python 2.3 버전이 설치되어 있으므로 --nodeps 옵션을 사용해서 설치한다.

5.2 환경 설정

프로그램 실행에 필요한 library path를 지정해 준다.

```
export PYTHONPATH=/usr/lib/python2.4/site-packages
```

scmsweb-gridway MAD 파일을 Gridway 실행 디렉토리로 복사한 후, 수정한다.

```
rpm -qif scmsweb-gridway-3.2.2-4
cp /usr/share/scmsweb/gw_im_mad_scmsweb $GW_LOCATION/bin
vi $GW_LOCATION/bin/gw_im_mad_scmsweb
```

SCMSWEB_GW_IM="python /usr/lib/python2.4/site-packages/scmsweb/gridway.py"
=>
SCMSWEB_GW_IM="python /usr/lib/python2.4/site-packages/scmsweb/gridway.py -u http://goc.pragma-grid.net/cgi-bin/scmsweb/xml_display.cgi?grid=on"

gridway configuration 파일에서 사용할 설정 파일들을 $GW_LOCATION/etc 밑에 만든다.

```
cd $GW_LOCATION/etc
```

$GW_LOCATION/etc/gw_scmsweb_prews.conf

```
[main]
grid=PRAGMA
#ignore_hostlist=sakura.hpcc.jp,rocks-153.sdsc.edu
ignore_hostlist=pragma1.grid.hku.hk,sakura.hpcc.jp,rocks-153.sdsc.edu,cafe01.exp-net
.osaka-u.ac.jp,pragma.lzu.edu.cn
#--------------------------------------------------
# [filter]
# #bandwidth=5
# self=1
# gram=prewsgram
# onegram_only=1
# probe=IntelCompiler
#--------------------------------------------------
```

$GW_LOCATION/etc/gw_scmsweb_ws.conf

```
[main]
grid=PRAGMA
```

$GW_LOCATION/etc/pragma_ignore_hostlist

```
pragma1.grid.hku.hk
sakura.hpcc.jp
rocks-153.sdsc.edu
cafe01.exp-net.osaka-u.ac.jp
pragma.lzu.edu.cn
```

※ 작업 실행에 문제가 있어 제외시키고자 하는 호스트 이름을 적어놓은 파일

Gridway Configuration 파일에 다음 내용을 추가한다.

$GW_LOCATION/etc/gwd.conf

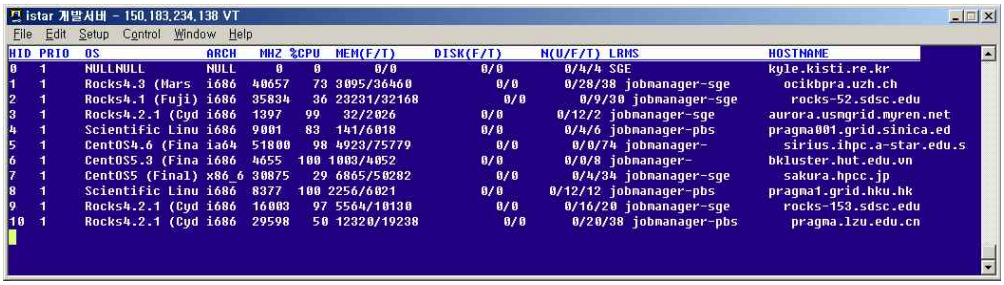
```
# scms
IM_MAD = scms_prews:gw_im_mad_scmsweb:-b /usr/local/gw/etc/gw_scmsweb_prews.conf:gridftp:prews
#IM_MAD = scms_gridftp:gw_im_mad_scmsweb:-h /usr/local/gw/etc/im_examples/myhost.list:gridftp:prews
IM_MAD = scms_ws:gw_im_mad_scmsweb:-b /usr/local/gw/etc/gw_scmsweb_ws.conf:gridftp:ws
EM_MAD = prews:gw_em_mad_prews::rsl
EM_MAD = ws:gw_em_mad_ws::rsl2
TM_MAD = gridftp:gw_tm_mad_ftp:
```

마지막으로 /etc/sce/sce.conf 파일에서 [scmsweb] 섹션을 다음과 같이 변경한다.

```
[scmsweb]
refresh_time = 60
clustername = GOC
pointer_url = localsite
webadmin_email = Do-not-notify
url = http://kyle01.kisti.re.kr
lock_file = /tmp/scmsweb_xml.lock
org = TNGC_TH
cgiurl = /cgi-bin/scmsweb
gridname = PRAGMA
tcp_gettimeout = 30
support_email = hgkim@kisti.re.kr
```

5.3 PRAGMA 자원 연동화면

다음은 gwhost -c 2 명령으로 PRAGMA 자원의 현재 상황을 본 화면이다.



```
istar 개발서버 - 150.183.234.138 VT
File  Edit  Setup  Control  Window  Help
HID PRIO  OS              ARCH    MHZ %CPU  MEM(F/T)      DISK(F/T)      N(U/F/T) LRMS                  HOSTNAME
0   1     NULLNULL        NULL      0    0        0/0           0/0        0/4/4 SGE                   kyle.kisti.re.kr
1   1     Rocks4.3 (Mars  i686  40657   73 3095/36460           0/0       0/28/38 jobmanager-sge         ocikbpra.uzh.ch
2   1     Rocks4.1 (Fuji) i686  35834   36 23231/32168          0/0        0/9/30 jobmanager-sge         rocks-52.sdsc.edu
3   1     Rocks4.2.1 (Cyd i686  1397    99   32/2026           0/0       0/12/2 jobmanager-sge        aurora.usmgrid.myren.net
4   1     Scientific Linu i686  9001   83  141/6018           0/0        0/4/6 jobmanager-pbs        pragma001.grid.sinica.ed
5   1     CentOS4.6 (Fina ia64  51800   98 4923/75779           0/0        0/0/74 jobmanager-           sirius.inpc.a-star.edu.s
6   1     CentOS5.3 (Fina i686  4655  100 1003/4052            0/0        0/0/8 jobmanager-           bkluster.hut.edu.vn
7   1     CentOS5 (Final) x86_6 30875   29 6865/50282           0/0        0/4/34 jobmanager-sge         sakura.hpcc.jp
8   1     Scientific Linu i686  8377  100 2256/6021            0/0       0/12/12 jobmanager-pbs        pragma1.grid.hku.hk
9   1     Rocks4.2.1 (Cyd i686  16003   97 5564/10130           0/0       0/16/20 jobmanager-sge         rocks-153.sdsc.edu
10  1     Rocks4.2.1 (Cyd i686  29598   50 12320/19238          0/0       0/20/38 jobmanager-pbs         pragma.lzu.edu.cn
```

※ Gridway 가 제대로 동작하지 않을 경우, $GW_LOCATION/var/gwd.log 파일을 살펴보고, 동작하지 않는 이유를 살펴본다.

6. Ganga

6.1 Ganga 설치

파이썬으로 작성된 설치 스크립트를 내려 받은 후, 실행한다.

http://ganga.web.cern.ch/ganga/download/ganga-install

설치는 개인 사용자마다 설치해서 사용할 수도 있고, 공용 디렉토리에 설치하고 함께 사용할 수도 있다.

공용으로 설치할 경우 다음과 같이 설치하면 된다.

```
mkdir -p /opt/ganga
python ganga-install --prefix=/opt/ganga LAST
```

여기서는 LAST 옵션으로 가장 최신 버전을 설치하였으나, 특정 버전이 필요한 경우에는 LAST 대신에 버전 번호를 적어주면 된다.

6.2 Ganga Configuration 파일 설정

/opt/ganga/install/5.4.0/bin/ganga -g 명령으로 .gangarc 파일을 생성하고, 다음 사항들을 추가해 준다.

GangaKISTI 플러그인을 사용할 수 있도록 RUNTIME_PATH 다음과 같이 설정한다.

```
RUNTIME_PATH = GangaKISTI
```

※ 두 개 이상의 플러그인을 함께 사용할 경우 콜론(:)으로 구분해서 적어준다.

RUNTIME_PATH = GangaKISTI:GangaTutorial

GLITE 미들웨어 관련 옵션을 설정한다.

```
#  LCG/gLite/EGEE configuration parameters
[LCG]
GLITE_ENABLE = True
VirtualOrganisation = fkppl.kisti.re.kr
#  sets the default storage element
DefaultSE = ccsrm02.in2p3.fr
```

```
[defaults_LCG]
middleware = GLITE
```

voms 설정

```
#  default attribute values for GridProxy objects
[defaults_GridProxy]
voms = fkppl.kisti.re.kr
```

Sandbox 관련 SE 설정

```
[defaults_LCGSandboxCache]
se = ccsrm02.in2p3.fr
```

Sandbox 최대 크기 지정 (BoundSanboxLimit를 넘어갈 경우, 자동적으로 SE를 이용해서 동작한다.)

```
BoundSandboxLimit = 102400
```

특정 CE를 사용하도록 지정하기

```
AllowedCEs = darthvader.kisti.re.kr:2119/jobmanager-lcgpbs-fkppl
```

※ 사용 가능한 CE의 목록은 다음 명령으로 알 수 있다.

lcg-infosites --vo fkppl.kisti.re.kr ce

6.3 Ganga 실행

● VOMS Certificate 만들기

Ganga를 실행하기 전에 voms-proxy-init 명령으로 grid 상에서 사용할 certificate를 먼저 만들어 준다.

```
$ voms-proxy-destroy
$ voms-proxy-init --voms fkppl.kisti.re.kr
Enter GRID pass phrase:
Your identity: /C=KR/O=KISTI/O=GRID/O=KISTI/CN=55879660 HanGi kim
Creating temporary proxy ..................................................... Done
Contacting palpatine.kisti.re.kr:15000
[/C=KR/O=KISTI/O=GRID/O=eScience/CN=host/palpatine.kisti.re.kr] "fkppl.kisti.re.kr"
Done
Creating proxy ............................ Done
Your proxy is valid until Tue Nov 24 02:09:35 2009
```

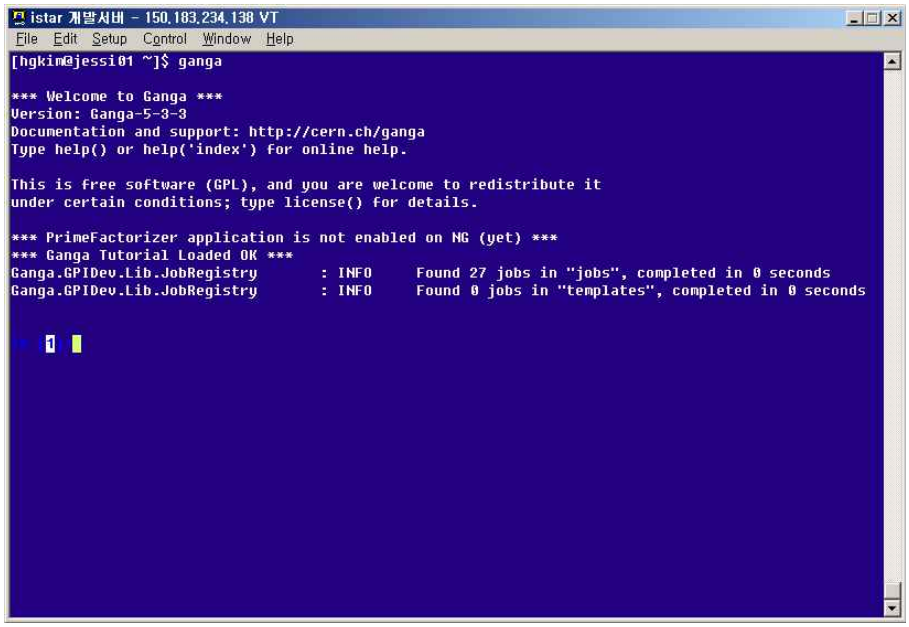
certificate가 만들어지면 voms-proxy-info 명령으로 만들어진 certificate를 확인한다.

```
$ voms-proxy-info -all
subject   : /C=KR/O=KISTI/O=GRID/O=KISTI/CN=55879660 HanGi kim/CN=proxy
issuer    : /C=KR/O=KISTI/O=GRID/O=KISTI/CN=55879660 HanGi kim
identity  : /C=KR/O=KISTI/O=GRID/O=KISTI/CN=55879660 HanGi kim
type      : proxy
strength  : 1024 bits
path      : /tmp/x509up_u500
timeleft  : 11:59:23
=== VO fkppl.kisti.re.kr extension information ===
VO        : fkppl.kisti.re.kr
subject   : /C=KR/O=KISTI/O=GRID/O=KISTI/CN=55879660 HanGi kim
issuer    : /C=KR/O=KISTI/O=GRID/O=eScience/CN=host/palpatine.kisti.re.kr
attribute : /fkppl.kisti.re.kr/Role=NULL/Capability=NULL
timeleft  : 11:59:07
uri       : palpatine.kisti.re.kr:15000
```

Ganga는 command line 모드와 gui 모드로 동작 가능하다.

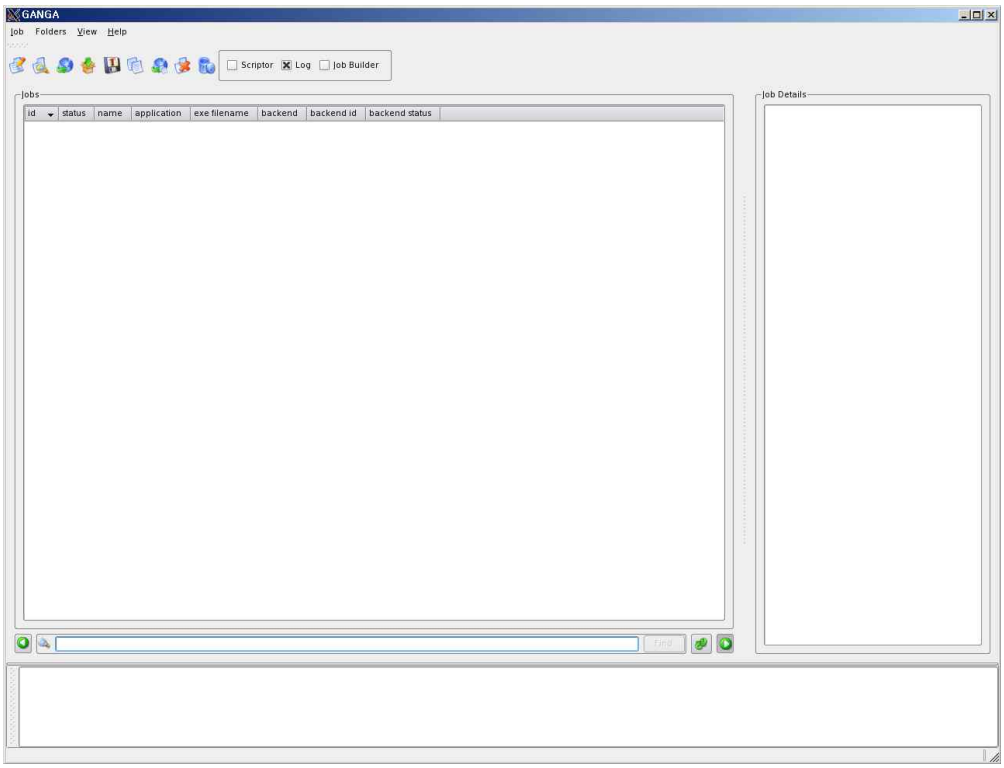
● Command line 모드

$ ganga



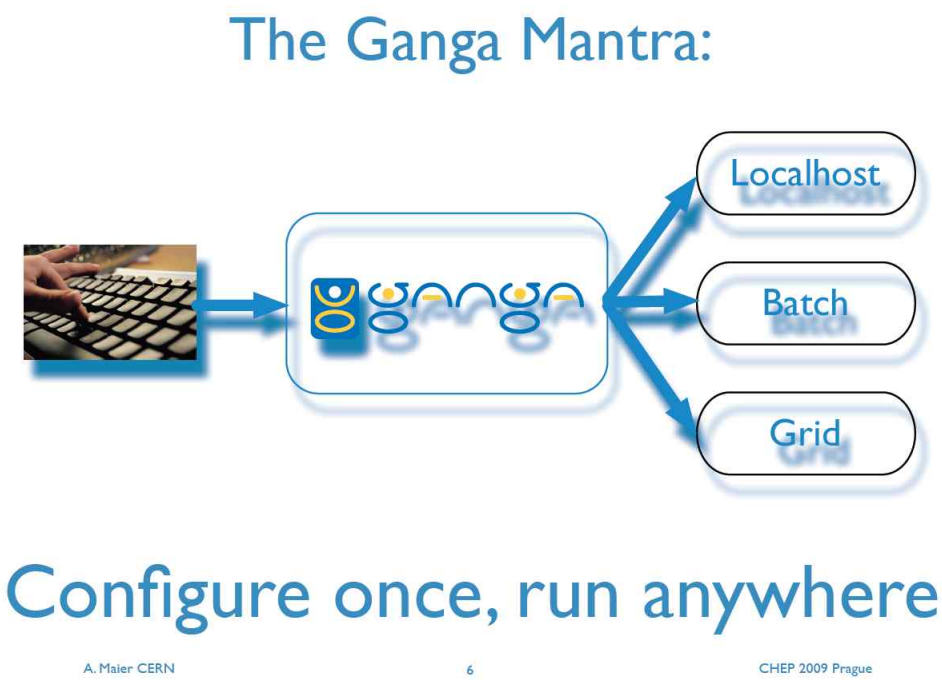
● GUI line 모드

$ ganga --gui



III. Ganga를 이용한 3개 이기종 인프라 활용방법

이번 장에서는 Ganga 백엔드(Backend)로 EGEE, PRAGMA, KISTI 슈퍼컴퓨터를 사용하는 방법에 대해서 알아본다.



Ganga는 위의 그림에서 알 수 있듯이 동일한 스크립트를 백엔드(backend)만 바꿔줌으로써 여러 인프라에서 동작할 수 있도록 지원해 준다.

간단한 Hello World 예제에 대해서 살펴 본 후, 입출력을 사용하는 stub 프로그램을 사용해서 각 인프라를 어

떻게 사용하면 되는지 살펴보도록 하겠다. 그리고 마지막으로 Autodock Application 플러그인(plug-in)의 사용 방법에 대해서 살펴보도록 한다.

1. EGEE 인프라

1.1 Hello World Locally

```
j=Job()
j.backend=Local()
j.submit()
```

1.2 Hello World on EGEE

```
j=Job()
j.backend=LCG()
j.backend.middleware = 'GLITE'
j.submit()
```

glite-WMS를 사용해서 EGEE 그리드 상에서 작업을 실행하기 위해서는 다음 설정 사항이 되어 있어야 한다.

In [1]:config['LCG']['GLITE_ENABLE'] = True

In [8]:j.backend.middleware = 'GLITE'

※ 이 사항은 Ganga 설정파일(.gangarc)에서도 설정할 수 있다.

1.3 입출력을 사용하는 stub 프로그램

파일 입출력을 사용하는 예제를 위해 다음의 stub C++ 프로그램을 사용한다.

stub.cpp

```
#include <iostream>
#include <fstream>
#include <unistd.h>

using namespace std;

int main()
{
    ifstream input("input_data.txt");
    string str;
    getline(input, str);

    cout << str << endl;

    ofstream outfile("output_data.txt");
    outfile << "This is Ganga output test" << endl;
    outfile.close();

    cout << "begin sleep" << endl;
    sleep(30);
    cout << "end sleep" << endl;

    return 0;
}
```

stub 프로그램은 input_data.txt 파일의 내용을 읽어서 화면에 출력하고, 지정된 문자열을 output_data.txt 파일로 출력한다.

input_data.txt

```
This is Ganga input test
```

stub.cpp 파일을 다음과 같이 컴파일해서 실행파일을 만든다.

```
$g++ -Wall -o stub stub.cpp
```

ex01.py

```
j = Job()
j.application=Executable()
j.application.exe=File('stub')
j.application.args=[]
j.inputsandbox=['input_data.txt']
j.outputsandbox=['output_data.txt']
j.backend = LCG()
j.submit()
```

작성된 Ganga 스크립트는 execfile 명령으로 실행할 수 있다.

```
execfile('ex01.py')
Ganga.GPIDev.Lib.Job : INFO submitting job 108
Ganga.GPIDev.Adapters : INFO submitting job 108 to LCG backend
Ganga.GPIDev.Lib.Job : INFO job 108 status changed to "submitted"
```

작업이 실행이 완료되면 Job의 peek 명령을 사용해서 실행결과를 확인할 수 있다.

peek(self, filename='', command='') method of Ganga.GPIDev.Lib.Job.Job.Job instance

Allow viewing of job output (and input) files

```
j=jobs[-1]
j.peek()
total 52
-rw-r--r--  1 hgkim hgkim   26 Nov 23 13:57 output_data.txt
-rw-rw-r--  1 hgkim hgkim 8039 Nov 23 14:02 __jobloginfo__.log
-rw-rw-r--  1 hgkim hgkim  156 Nov 23 14:23 _output_sandbox.tgz
-rw-rw-r--  1 hgkim hgkim   69 Nov 23 14:23 stdout.gz
-rw-rw-r--  1 hgkim hgkim   29 Nov 23 14:23 stderr.gz
-rw-rw-r--  1 hgkim hgkim 1953 Nov 23 14:23 __jobscript__.log
j.peek('stdout.gz', 'zcat')
This is Ganga input test
begin sleep
end sleep
j.peek('output_data.txt')
This is Ganga output test
/home/hgkim/gangadir/workspace/hgkim/LocalAMGA/108/output/output_data.txt (END)
```

LCG 그리드에 작업이 제출되면

glite-wms-job-logging-info 명령으로 작업 진행 상황을 확인할 수 있다.

```
j=jobs(108)
cat $j.backend.loginfo(verbosity=2)
```

```
*****************************************************************
LOGGING INFORMATION:

Printing info for the Job : https://snow.kisti.re.kr:9000/8SFY2mqkBbvx7r7ULaMqzA

        ---
Event: RegJob
- Arrived                    =    Mon Nov 23 13:53:10 2009 KST
- Host                       =    snow.kisti.re.kr
-   Ns                                                          =
https://150.183.250.208:7443/glite_wms_wmproxy_server
- Nsubjobs                   =    0
- Source                     =    NetworkServer
-    Src    instance                                            =
https://150.183.250.208:7443/glite_wms_wmproxy_server
- Timestamp                  =    Mon Nov 23 13:53:10 2009 KST
- User                       =    /C=KR/O=KISTI/O=GRID/O=KISTI/CN=55879660
HanGi kim
... 중간 생략
Event: Done
- Arrived                    =    Mon Nov 23 14:23:49 2009 KST
- Exit code                  =    0
- Host                       =    snow.kisti.re.kr
- Reason                     =    Job terminated successfully
- Source                     =    LogMonitor
- Src instance               =    unique
- Status code                =    OK
- Timestamp                  =    Mon Nov 23 14:23:49 2009 KST
- User                       =    /C=KR/O=KISTI/O=GRID/O=KISTI/CN=55879660
HanGi kim/CN=proxy/CN=proxy
        ---
Event: Clear
- Arrived                    =    Mon Nov 23 14:23:55 2009 KST
- Host                       =    snow.kisti.re.kr
- Reason                     =    USER
- Source                     =    NetworkServer
- Src instance               =    28651
- Timestamp                  =    Mon Nov 23 14:23:55 2009 KST
-   User                                                        =
/C=KR/O=KISTI/O=GRID/O=eScience/CN=host/snow.kisti.re.kr/CN=1085207265

*****************************************************************
```

1.4 Autodock Application 플러그인

autodock.py

```
j=Job()
j.application=Autodock()
j.application.exe="/bin/sh"
j.application.script="/home/kyle/ganga_autodock/autodock.sh"
j.application.binary="/home/kyle/ganga_autodock/autodock.tar.gz"
j.application.protein="/home/kyle/ganga_autodock/1u2y.tar.gz"
j.application.ligand="/home/kyle/ganga_autodock/ligands/9004736_1.pdbq"
j.application.parameter="/home/kyle/ganga_autodock/dpf3gen.awk"
j.backend=LCG()
j.submit()
```

autodock.py 스크립트의 실행을 위해서는

autodock.sh : 실행 스크립트

autodock.tar.gz : autodock3 프로그램

1u2y.tar.gz : 단백질 정보 파일

9004736_1.pdbq : 리간드 정보 파일

dpf3gen.awk : awk 스크립트

파일이 필요하다.

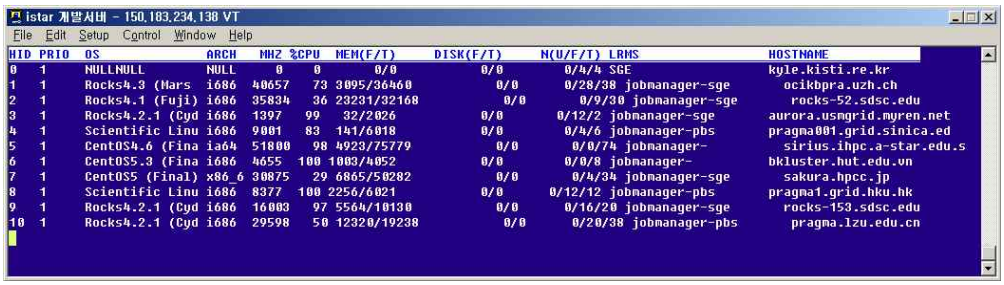
autodock.sh

```
#!/bin/sh
if [ $# -lt 3 ]
    then
        echo "Usage: $0 <protein.tar.gz> <ligand.pdbq> <dpf3gen.awk>"
        echo "  e.g. $0 1u2y.tar.gz 9004736_1.pdbq dpf3gen.awk"
        exit 1
fi

protein_tar_file=$1
ligand_file=$2
dpf3gen_file=$3
protein_name=${protein_tar_file%.tar.gz}
ligand_name=${ligand_file%.pdbq}
export SHELL=/bin/sh
export AUTODOCK_UTI=$PWD
export LD_LIBRARY_PATH=$PWD:$LD_LIBRARY_PATH
ulimit -s unlimited

echo $LD_LIBRARY_PATH

tar xzf autodock.tar.gz
tar zxf $protein_tar_file
cp $protein_name/* .
./mkdpf3 $ligand_file $protein_name.pdbqs
./autodock3 -p $ligand_name.$protein_name.dpf -l $ligand_name₩_$protein_name.dlg
true
```

2. PRAGMA

2.1 PRAGMA 그리드 상태 확인하기

PRAGMA에 작업을 제출하기 전에 gwhost 명령으로 PRAGMA 그리드의 상태가 어떤지 확인한다.



```
istar 개발서버 - 150.183.234.138 VT
File  Edit  Setup  Control  Window  Help
HID PRIO  OS              ARCH    MHZ %CPU  MEM(F/T)      DISK(F/T)      N(U/F/T) LRMS                HOSTNAME
0   1     NULLNULL        NULL      0    0        0/0          0/0        0/4/4 SGE                  kyle.kisti.re.kr
1   1     Rocks4.3 (Mars  i686   40657   73 3095/36460          0/0       0/28/38 jobmanager-sge        ocikbpra.uzh.ch
2   1     Rocks4.1 (Fuji) i686   35834   36 23231/32168         0/0        0/9/30 jobmanager-sge         rocks-52.sdsc.edu
3   1     Rocks4.2.1 (Cyd i686   1397    99   32/2026          0/0       0/12/2 jobmanager-sge        aurora.usmgrid.myren.net
4   1     Scientific Linu i686   9001    83  141/6018          0/0        0/4/6 jobmanager-pbs        pragma001.grid.sinica.ed
5   1     CentOS4.6 (Fina ia64   51800   98 4923/75779          0/0        0/0/74 jobmanager-           sirius.ihpc.a-star.edu.s
6   1     CentOS5.3 (Fina i686   4655   100 1003/4052          0/0        0/0/8 jobmanager-           bkluster.hut.edu.vn
7   1     CentOS5 (Final) x86_6 30875    29 6865/50282          0/0        0/4/34 jobmanager-sge        sakura.hpcc.jp
8   1     Scientific Linu i686   8377   100 2256/6021          0/0      0/12/12 jobmanager-pbs        pragma1.grid.hku.hk
9   1     Rocks4.2.1 (Cyd i686   16003   97 5564/10130          0/0       0/16/20 jobmanager-sge        rocks-153.sdsc.edu
10  1     Rocks4.2.1 (Cyd i686   29598   50 12320/19238         0/0       0/20/38 jobmanager-pbs         pragma.lzu.edu.cn
```

2.2 Hello World on PRAGMA

작업이 실행될 백엔드를 EGEE에서 PRAGMA로 변경하기 위해서는 다음과 같이 Job의 backend 필드만 변경해 주면 된다.

```
j=Job()
j.backend=Gridway()
j.submit()
```

2.3 Helo World on EGEE or PRAGMA

InterGrid 백엔드를 사용하면 LCG 백엔드와 Gridway 백엔드 중에서 CPU 작업 로드가 적은 쪽으로 자동적으로 작업이 제출된다.

```
j=Job()
j.backend=InterGrid()
j.backend.targetBackends=["LCG","Gridway"]
j.submit()
```

2.4 입출력을 사용하는 stub 프로그램

```
j = Job()
j.application=Executable()
j.application.exe=File('stub')
j.application.args=[]
j.inputsandbox=['input_data.txt']
j.outputsandbox=['output_data.txt']
j.backend = Gridway()
j.submit()
```

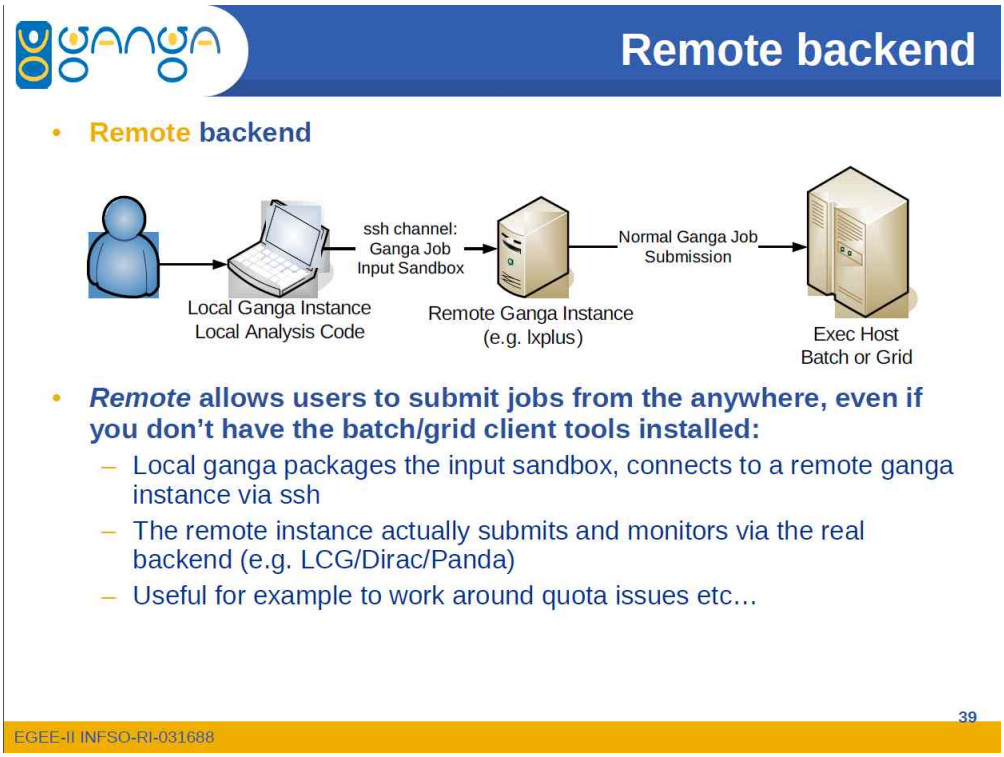
```
j = Job()
j.application=Executable()
j.application.exe=File('stub')
j.application.args=[]
j.inputsandbox=['input_data.txt']
j.outputsandbox=['output_data.txt']
j.backend = InterGrid()
j.backend.targetBackends=["LCG","Gridway"]
j.submit()
```

2.5 Autodock Application 플러그인

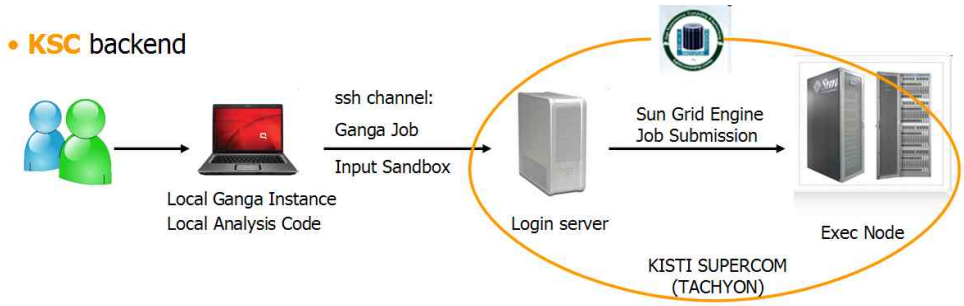
```
j=Job()
j.application=Autodock()
j.application.exe="/bin/sh"
j.application.script="/home/kyle/ganga_autodock/autodock.sh"
j.application.binary="/home/kyle/ganga_autodock/autodock.tar.gz"
j.application.protein="/home/kyle/ganga_autodock/1u2y.tar.gz"
j.application.ligand="/home/kyle/ganga_autodock/ligands/9004736_1.pdbq"
j.application.parameter="/home/kyle/ganga_autodock/dpf3gen.awk"
j.backend = InterGrid()
j.backend.targetBackends=["LCG","Gridway"]
j.submit()
```

3. KISTI 슈퍼컴퓨터

KISTI 슈퍼컴퓨터 Tachyon을 Ganga의 백엔드로 사용할 수 있도록 해주는 KSC 백엔드 플러그인은 영국 BIRMINGHAM 대학 물리학과의 Mark Slater가 만든 Remote 백엔드를 기반으로 만들어졌다.



KSC 백엔드는 Remote 호스트에 있는 SGE(Sun Grid Engine)에서 작업을 실행할 수 있도록 해주며 다음과 같이 동작한다.



3.1 Hello World on Remote SGE

```
j=Job()
j.backend = KSC()
j.backend.host = "kyle.kisti.re.kr"
j.backend.username = "globus"
j.backend.queue_name = "all.q"
j.backend.ganga_dir = "/home/globus/gangadir/remote_jobs"
j.submit()
```

j.backend.ganga_dir 필드에는 remote 작업을 실행하고, 모니터링 하는데 필요한 스크립트 및 작업 파일이 저장될 디렉토리를 지정해 준다.

j.backend.queue_name 필드에는 작업을 실행할 SGE queue 이름을 적어 준다.

다음은 작업 실행 상태를 확인하는 화면으로(jobs 명령) name 필드에서는 SGE job id를, backend.actualCE 필드에서는 실제 작업이 실행되는 호스트 이름과 queue 이름을 알 수 있다.

84 completed 718 Executable KSC kyle.kisti.re.kr/all.q

3.2 입출력을 사용하는 stub 프로그램 on Remote SGE

```
j=Job()
j.application=Executable()
j.application.exe=File('stub')
j.application.args=[]
j.inputsandbox=['input_data.txt']
j.outputsandbox=['output_data.txt']
j.backend = KSC()
j.backend.host = "kyle.kisti.re.kr"
j.backend.username = "globus"
j.backend.ganga_dir = "/home/globus/gangadir/remote_jobs"
j.submit()
```

In [33]:jobs

85 submitted 719 Executable KSC kyle.kisti.re.kr/all.q

```
j.peek()
total 12
-rw-r--r--  1 hgkim gwusers  0 Nov 23 15:39 __syslog__
-rw-r--r--  1 hgkim gwusers 47 Nov 23 15:39 stdout
-rw-r--r--  1 hgkim gwusers  0 Nov 23 15:39 stderr
-rw-r--r--  1 hgkim gwusers 26 Nov 23 15:39 output_data.txt
-rw-r--r--  1 hgkim gwusers 86 Nov 23 15:39 __jobstatus__
```

```
j.peek('stdout')
This is Ganga input test
begin sleep
end sleep
/home/hgkim/gangadir/workspace/hgkim/LocalAMGA/85/output/stdout (END)
```

```
j.peek('output_data.txt')
This is Ganga output test
/home/hgkim/gangadir/workspace/hgkim/LocalAMGA/85/output/output_data.txt (END)
```

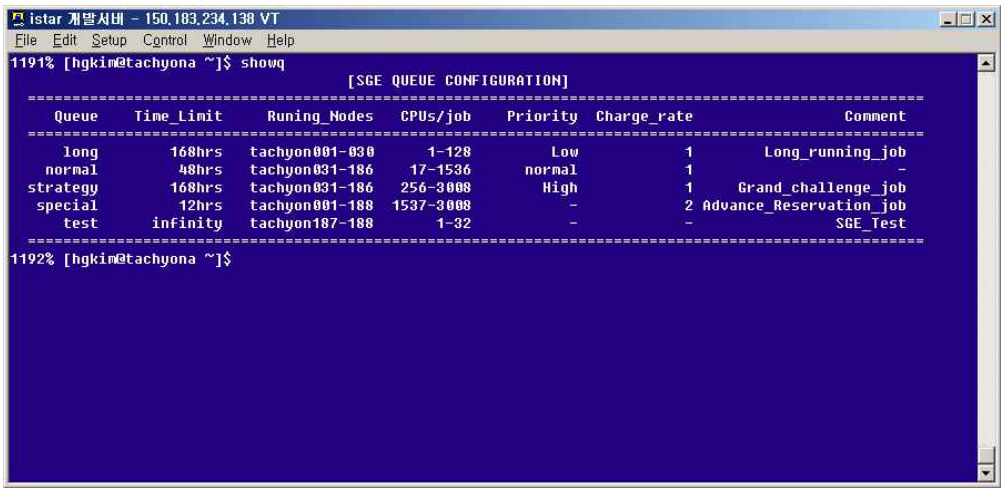
3.3 Hello World on KISTI TachyonII

```
j=Job()
j.backend = KSC()
j.backend.host = "tachyon.ksc.re.kr"
j.backend.username = "hgkim"
j.backend.ganga_dir = "/work01/hgkim/remote_jobs"
j.backend.queue_name= "long"
j.backend.pre_script = ['cd /work01/hgkim/']
j.backend.subprocess_dir =
"/work01/hgkim/Ganga/external/subprocess/2.4.2/noarch/lib/python2.2/site-packages"
j.submit()
```

j.backend.ganga_dir 필드에는 remote 작업을 실행하고, 모니터링 하는데 필요한 스크립트 및 작업 파일이 저장될 디렉토리를 지정해 준다.

j.backend.queue_name 필드에는 작업을 실행할 SGE queue 이름을 적어 준다. 일반적인 사용자는 long queue 를 사용하면 된다.

참고로 TachyonII에는 showq 명령으로 사용가능한 queue 를 확인할 수 있다.



j.backend.pre_script 필드에는 qsub 명령으로 작업을 제출하기 전에 실행할 사항들을 적어준다. TachyonII에서는 작업 제출을 스크래치 디렉토리(/work01혹은 /work02)에서 하도록 되어 있으므로 스크래치 디렉토리로 변경하는 명령을 이 필드에 적어준다.
Ganga 프로그램을 실행하기 위해서는 python subprocess 패키지가 필요하다. 일반적으로 subprocess 패키지가 없을 경우에는 CERN의 Ganga 웹 서버에서 필요한 파일을 가져오나, TachyonII의 로그인서버에는 밖으로 나가는 것이 금지되어 있으므로 스크래치 디렉토리에 필요한 파일을 복사해 놓고 사용하다.
j.backend.subprocess_dir필드에는 subprocess 패키지가 있는 디렉토리를 지정해준다.

다음은 작업 실행 상태를 확인하는 화면으로(jobs 명령) name 필드에서는 SGE job id를, backend.actualCE 필드에서는 실제 작업이 실행되는 호스트 이름과 queue 이름을 알 수 있다.

```
# 89 completed 414277 Executable
KSC tachyon.ksc.re.kr/long
```

3.4 입출력을 사용하는 stub 프로그램 on KISTI TachyonII

```
j = Job()
j.application=Executable()
j.application.exe=File('stub')
j.application.args=[]
j.inputsandbox=['input_data.txt']
j.outputsandbox=['output_data.txt']
j.backend = KSC()
j.backend.host = "tachyon.ksc.re.kr"
j.backend.username = "hgkim"
j.backend.ganga_dir = "/work01/hgkim/remote_jobs"
j.backend.queue_name= "long"
j.backend.pre_script = ['cd /work01/hgkim/']
j.backend.subprocess_dir =
"/work01/hgkim/Ganga/external/subprocess/2.4.2/noarch/lib/python2.2/site-packages"
j.submit()
```

In [35]:jobs

94 running 414283 Executable KSC tachyon.ksc.re.kr/long

```
j.peek()
total 12
-rw-r--r--  1 hgkim gwusers  0 Nov 23 16:14 __syslog__
-rw-r--r--  1 hgkim gwusers 47 Nov 23 16:14 stdout
-rw-r--r--  1 hgkim gwusers  0 Nov 23 16:14 stderr
-rw-r--r--  1 hgkim gwusers 26 Nov 23 16:14 output_data.txt
-rw-r--r--  1 hgkim gwusers 86 Nov 23 16:14 __jobstatus__
```

```
.peek('stdout')
This is Ganga input test
begin sleep
end sleep
/home/hgkim/gangadir/workspace/hgkim/LocalAMGA/95/output/stdout (END)
```

```
j.peek('output_data.txt')
This is Ganga output test
/home/hgkim/gangadir/workspace/hgkim/LocalAMGA/95/output/output_data.txt (END)
```

1. 본 책자는 한국과학기술정보연구원의 2009년도 "e-Science 환경구축" 과제로 수행한 연구결과입니다.
2. 본 책자의 내용을 발표할 때에는 반드시 한국과학기술정보연구원의 연구결과임을 밝혀야 합니다.