Computing grids

Yiannis Georgiou

SC-CAMP school

18 august 2010

CIGRI lightweight Grid Technical Exercises

Overview

In this exercise we will :

- learn the basics of the function of CIGRI lightweight grid.
- deploy a personal CIGRI grid upon Grid5000 platform
- run simple multiparametric applications
- program our own multiparametric bag-of-tasks application to find a secret key to open a particular file
- learn how to collect results

Connection to a Grid5000 frontal

- Use your personal username and password and open an `ssh` connection to server `digitalis.imag.fr`

```
$ ssh sccamp1@digitalis.imag.fr
sccamp1@digitalis.imag.fr's password:

sccamp1@digitalis:~$
```

Connection to a Grid5000 computing node

- Open an ssh connection to the particular node that is given to you gdx128.orsay.grid5000.fr with :
  username : g5k
  password : grid5000

```
$ ssh g5k@gdx-128.orsay.grid5000.fr
Password:
g5k@gdx-128:~
```

- Verification of the file /tmp/hosts that has the particular nodes that will be deployed as the grid.
- Every node will get a particular role in your personal grid.

```
$ cat /tmp/hosts
gdx-128.orsay.grid5000.fr
...
```

Start CIGRI

- Use the following command which will deploy the different services upon each node in order to have a CIGRI grid ready for utilization.

```
$ /home/g5k/init_cigri_oar.sh -f /tmp/hosts -n2
```

- After some seconds the roles attribution and services deployment will finish. Verify the file `/home/g5k/env.sh` and note the names of the CIGRI server and the OAR cluster servers.

```
$ cat /home/g5k/env.sh
#!/bin/sh
export CIGRI_SERVER=gdx-128.orsay.grid5000.fr
export CIGRI_CLUSTER0=gdx-129.orsay.grid5000.fr
export CIGRI_CLUSTER1=gdx-132.orsay.grid5000.fr
```

Verify CIGRI database

- The database has a central role for CIGRI and all different informations for jobs,clusters,resources,errors, etc are stored upon it. Connect to the database :

```
$ sudo mysql
```

- select the database cigri

```
$ use cigri;
```

- and check out the various information that have been stored :

```
> show tables;
> select * from clusters;
> select * from nodes;
```

CIGRI commands

- The following CIGRI commands may be used for grid jobs submission :

```
$ gridsub
```

- monitoring of jobs

```
$ gridstat
```

- jobs deletion

```
$ griddel
```

Submit and follow a simple grid job

In this exercise we learn how to submit, monitor and collect results of a simple grid job upon CIGRI. The procedure is composed by the following steps :

- Preparation of the multi-parametric grid job (simple bag-of-tasks application) for submission upon CIGRI.
- Submission of the grid job upon CIGRI with the use of `gridsub` command
- Monitoring of the job in the grid with the use of `gridstat` command
- Collection of the results once the job is finished through the module `Collector`
- Extraction and verification of results.

- Go to the directory : /home/g5k/SC-CAMP2010_Tutorial_OAR-CIGRI/Exercise_1st/

```
$ cd /home/g5k/SC-CAMP2010_Tutorial_OAR-CIGRI/Exercise_1st/
```

A CIGRI grid job (bag-of-tasks) is composed by :

- a **script file** written in jdl language which resides upon the CIGRI server and represents the actual CIGRI grid job.
- an *executable program* (any possible format), which resides upon the CIGRI-OAR clusters and represents an independent task (of the grid job)
- and a **file of parameters** which resides upon the CIGRI server and each line of parameters is used as argument to the *executable program* to compose an independent task.

- Open and edit the jdl file to fit the needs of your personal CIGRI. In particular you need to change the cluster names and in place of `cluster1` and `cluster2` you have to put the names of the OAR cluster servers.

```
DEFAULT{
  name = benchmark_sleep;
  paramFile = /home/g5k/SC-CAMP2010_Tutorial_OAR-CIGRI/Exercise_1st/sleep.param ;
}
gdx-129.orsay.grid5000.fr{
  execFile = /home/g5k/SC-CAMP2010_Tutorial_OAR-CIGRI/Exercise_1st/sleep.sh ;
  execDir = /home/g5k/SC-CAMP2010_Tutorial_OAR-CIGRI/Exercise_1st/ ;
  walltime = 14:00:00 ;
}
gdx-132.orsay.grid5000.fr{
  execFile = /home/g5k/SC-CAMP2010_Tutorial_OAR-CIGRI/Exercise_1st/sleep.sh ;
  execDir = /home/g5k/SC-CAMP2010_Tutorial_OAR-CIGRI/Exercise_1st/ ;
  walltime = 14:00:00 ;
}
```

- Open the executable script `sleep.sh` and edit it : add the command `hostname` so as to know at which node each single task is executed.
- Observe how the parameter is passed to the executable :

```
$ vi sleep.sh
date
sleep $1
date
hostname
```

- How many parameters are needed to fill all the nodes of the grid ? Consider that each task will be executed upon a single CPU.
- Open the parameters file : each line will be passed as argument to the executable. Edit the parameter files with more parameters so as to fill all the grid.

```
$ vi sleep.param
34
56
```

- Submit the grid job

```
$ gridsub -f benchmark_sleep.jdl
```

Monitor the grid job

- as a user through `gridstat`
- or as an administrator through `the database`

When the job finishes :

- Collect the results
- Follow the commands in the file `README_Collect_Results`

```
$cat README_Collect_Results
mkdir /tmp/cigri/
sudo /home/cigri/Collector/collectorCigri.rb
```

Extract the stored results and verify the results

```
$tar zxvf 1.tgz
```

The problem of password cracking

The exercise consists of cracking a password of a `.rar` file in order to extract a particular file `secret.rar`. We will use the method of *brute force attack* which consists of trying all different combinations of symbols in order to find it (http ://en.wikipedia.org/wiki/Brute_force_attack)

- The password is composed by numbers from 1 to 10000000.
- The executable takes as arguments 2 numbers and tries all the combinations from the smaller to the larger one.

1)What is the interest of using the grid for the password cracking ? 2)How should we use the grid for finding the password ?

In this exercise we will learn how to design and create a grid job to perform real computations in order to crack a password. The procedure is composed by the following steps :

- Design of the grid job : Understanding of the particular algorithm and creation of the parameters file.
- Submission, monitoring and collection of results according to the procedure in the 1st exercise.
- Extraction of the results and search for the found password among the different terminated jobs.

- Similarly with the first exercise prepare the grid jdl script `benchmark_crack.jdl` by adapting the `cluster1` and `cluster2` to your personal cluster.
- Understand the function of the executable program `crack-secret.sh` and edit the parameters file `crack.param` in order to crack the password.
- How many parameters should you include in the file so as to find the password as fast as possible ?