Advanced Jobs

DIRAC Project

Overview

- Parametric Jobs
- MPI Jobs
- Why APIs are important?
- Why advanced users prefer APIs?
- How it is done?
- What is local mode what is not?
- Tutorial exercises
- What do you have learned?

Parametric Jobs

- A parametric job consist in submission of a set of jobs where only one parameter makes the difference between each job.
- To define this parameter the attribute "Parameters" has to be defined in the JDL, it can take the values:
  - A list (strings or numbers)
  - Or, an integer, in this case the attributes *ParameterStart* and *ParameterStep* must be defined in order to create the list of values.

Parametric Job - JDL

```
Executable = "testParametricJob.sh";
JobName = ”Parametric_%s";
Arguments = "%s";
Parameters = 20;
ParameterStart = 0;
ParameterStep = 2;
StdOutput = "StdOut_%s";
StdError = "StdErr_%s";
InputSandbox = {"testJob.sh"};
OutputSandbox = {"StdOut_%s","StdErr_%s"};
```

Placeholder replaced by Parameter value for each job

MPI Jobs

- Message Passing Interface (MPI) is commonly used to handle the communications between tasks in parallel applications.
- Two versions and implementations supported in DIRAC are:
  - MPICH-1 : MPICH1
  - MPICH-2 : MPICH2
- Users must know that, currently:
  - The jobs can only run on just one grid site. The maximum number of processors that a user can require for a job depends on the capacity of the sites.
  - Not all the sites are providing:
    - Shared home directories or SSH connection.

MPI - Jobs

- To submit MPI jobs in DIRAC:
  - As usual, define appropriate attributes in the JDL file to declare
    - MPI flavor
    - Number of CPUs required.
  - Create a wrapper script to manage the files between the nodes, depending of the site environment.

MPI Job - JDL

```
JobType        = "MPI";
CPUNumber      = 3;
Executable     = "mpi.sh";
Arguments      = "pearson_noblocking 3 ";
StdOutput      = "pearson_noblocking.out";
StdError       = "pearson_noblocking.err";
InputSandbox   = {"mpi.sh","pearson_noblocking.c”};
OutputSandbox  =
   {"pearson_noblocking.o","pearson_noblocking.err","pears
   on_noblocking.out"};
Flavor         = "MPICH2"
```

Why APIs are important?

- The DIRAC API is encapsulated in several Python classes designed to be used easily by users to access a large fraction of the DIRAC functionality. Using the API classes it is easy to write small scripts or applications to manage user jobs and data.
- The DIRAC API provides a transparent and secure way for users to submit jobs to the Grid.
- Permit debug locally the programs before submit jobs to the Grid.
- While it may be exploited directly by users, the DIRAC API also serves as the interface for the Ganga Grid front-end to perform distributed user analysis for LHCb.

Why advanced users prefer APIs

- API provides a simple, seamless interface to Grid resources to submit jobs that can be:
  - Single applications
  - Multiple steps of different applications
- Can perform a typical user analysis using understandable Python code

APIs

- The API allows creating DIRAC jobs using the Job object.
- Example:

```
from DIRAC.Interfaces.API.Job import Job
j = Job()
j.setExecutable('ls',arguments='-al',logFile='ls.log')
j.execute()
```

Note that all the DIRAC API commands described here may also be executed directly from the Python prompt.

Setting the computing resources to use:

- Operating system and system architecture:

```
j.setSystemConfig("slc4_ia32_gcc34")
```

- CPU requirement determined:

```
j.setCPUTime(21600)
```

- Site name:

```
j.setDestination('LCG.CERN.ch')
```

- Banned sites:

```
j.setBannedSites(['LCG.CNAF.it','LCG.CNAF-t2.it'])
```

APIs
Configurable Job Parameters

- Multiple Jobs:

```
for i in range(20):

    j.setName('MyJob_%d' % i)
```

- Log Level:

```
j.setLogLevel('DEBUG')
```

- Environment variables:

```
j.setExecutionEnv({'MYVARIABLE':'TOTO'})
```

APIs
Handling Data

- Input Sandbox

```
j.setInputSandbox(['Input.options','*.txt','lib/'])
```

- Output Sandbox

```
j.setOutputSandbox(['*.log','summary.data'])
```

- Input Data

```
j.setInputData(['/dirac/vo.formation.idgrilles.fr/
  user/h/hamar/input1.data'])
```

- Output Data

```
j.setOutputData(['output1.data','output2.data'],Outp
  utSE=['M3PEC-disk'],OutputPath='MyFirstAnalysis')
```

APIs Example

```
from DIRAC.Interfaces.API.Job
import Job j = Job() j.setName('MyFirstJob')
j.setOutputSandbox(['*.log','summary.data'])
j.setInputData(['/my/logical/file/name1',
                '/my/logical/file/name2'])
j.setOutputData(['output1.data','output2.data'],
                  OutputPath='MyFirstAnalysis')
j.setSystemConfig("slc4_ia32_gcc34")
j.setCPUTime(21600) j.setDestination('LCG.IN2P3.fr')
j.setBannedSites(['LCG.CNAF.it','LCG.CNAF-t2.it'])
j.setLogLevel('DEBUG')
j.setExecutionEnv({'MYVARIABLE':'TOTO'})
j.setExecutable('echo $MYVARIABLE')
print j._toJDL()
```

What is local mode?

- Local mode doesn't send the job to the Grid, the execution is not remote.
- The Local submission mode is a very useful tool to check the sanity of your job before submission to the Grid.
- The job executable is run locally in exactly the same way (same input, same output) as it will do on the Grid Worker Node.
- This allows to debug the job in a friendly local environment.

Local Mode Example

```
$ python
Python 2.5.5 (r255:77872, Mar 25 2010, 14:17:52)
[GCC 4.1.2 20080704 (Red Hat 4.1.2-46)] on linux2
Type "help", "copyright", "credits" or "license" for
more information.
>>> from DIRAC.Interfaces.API.Dirac import Dirac
>>> from DIRAC.Interfaces.API.Job import Job
>>> j = Job()
>>> j.setExecutable('/bin/echo hello')
{'OK': True, 'Value': ''}
>>> Dirac().submit(j,mode='local')
```

User exercises

- Exercises can be found at:

http://mareela.in2p3.fr:9200/dirac/wiki/DIRAC/Tutorials/JobManagementAdvanced

Note: All the files than you need are available as attachment of the wiki page!!!

What do you have learnt?

- How to submit Parametric and MPI Jobs.
- How to submit jobs using APIs.
- Debug the payload before submitting jobs to the grid
- Send multiples jobs
- Handle job management using APIs

Questions

Useful links

- http://dirac.in2p3.fr
- http://paterson.web.cern.ch/paterson/DIRAC3API_091208/DIRAC.Interfaces.API.Job.Job-class.html
- http://paterson.web.cern.ch/paterson/DIRAC3API_091208/DIRAC.Interfaces.API.Dirac.Dirac-class.html