Table of Contents

Grid and data management (Using ganga only)

What can you expect from the grid?

Prerequisites:

You will need to be familiar with python. After that you will need to do AT LEAST the following tutorials before starting this Ganga and Grid tutorial.

- You need a grid certificate
- Ganga Tutorial - Getting started with Ganga. Ganga Tutorial for LHCb, last updated 2009-03-16, using DaVinci v22r1
- DaVinciTutorial0 : **Introduction:** Running DaVinci for the first time, and the XMLSummarySvc.
- DaVinciTutorial0p5 : **Introduction:** Preparing the tutorials and a ganga job to run them.

You need at least one DaVinci job which works on the grid and makes some output files

Slides:

This tutorial corresponds to the slides last shown at the LHCb week here

1. Let's resubmit with changing things

I assume you have a previous successful job called `jobs(6)`, with a number of subjobs which we will copy and mess with below

Often, when you have submitted some jobs, a small number of them fails. The usual sucess rate of user grid jobs is 80%. The first thing to do is resubmit them, and see what that does.

If all of them failed, go back to testing locally, because it's probably a problem with your options or a mistake you made.

```
j=jobs(6)
failjobs= [js for js in j.subjobs if js.status=="failed"]


#just resubmit
for js in failjobs:
    js.resubmit()

#next try resubmitting with more CPU
for js in failjobs:
    js.backend.settings['CPUTime']=js.backend.settings['CPUTime']*2
    js.resubmit()

#Next try resubmitting to a different site
for js in failjobs:
    js.backend.settings['BannedSites']=[js.backend.actualCE]
    js.resubmit()

#next try making a new job, resplitting and resubmitting
j=j.copy()
j.inputdata=jobs('6.1').inputdata #copy a single failed job output
j.splitter.filesPerJob=1+len(j.inputdata)/10
j.submit()
```

Only if all the above doesn't work should you consider emailing the list. Ganga utilities (described below) can make this process much quicker.

2. Let's move some data

I assume you have a previous successful job called `jobs(7)` which we will copy and mess with below. I assume it makes a file called `DVnTuples.root`.

Often you will want to copy the data around, but think carefully before you do it. Most probably you can live with having files on the grid, and if you have to copy them somewhere, that doesn't have to be CERN.

```
#Let's send some data to grid storage
In [1]: j=jobs(7).copy()
In [2]: j.outputsandbox
Out[1]: ['DVHistos.root', 'DVnTuples.root']
In [3]: j.outputsandbox= ['DVHistos.root']
In [4]: j.outputdata=['DVnTuples.root']
In [5]: j.outputdata.location='GridTutorial'
In [6]: j.submit()
```

Then copy that data around

```
#first we need the list of files created
In [1]: ds=j.backend.getOutputDataLFNs()
#copy to RAL-USER, for example... not everything needs to be copied to CERN!
In [2]: ds.replicate('RAL-USER')
#download to your local hard disk
In [3]: ds[0].download('/tmp/')
In [4]: afile=PhysicalFile('/tmp/DVnTuples.root')
#upload it again
In [5]: dscp=afile.upload('/lhcb/user/<u>/<uname>/GridTutorial/DVnTuples.root')
#download the output into the job workspace, be sure you have enough space first!
In [6]: j.backend.getOutputData()
```

3. Where are the data?

```
#the outputsandbox is stored in the outputdir
In [1]: j.outputdir
#you can look at it like this
In [2]: j.peek()
#the outputdata may be anywhere on the grid
In [3]: ds=j.backend.getOutputDataLFNs()
In [4]: reps=ds.getReplicas()
In [5]: reps[ds[0].name]
#file 0 is replicated at these places
```

4. Using the Ganga Box

Before you delete the job, and if you want to keep the output for some time, why not put the LFNs in your Ganga Box?

```
In [1]: ds=j.backend.getOutputDataLFNs()
# syntax box.add(object, name_to_give_to_entry_in_box)
In [2]: box.add(ds, j.id+' '+j.name+' Output LFNs')
In [3]: j.remove()
In [4]: box #print the content of the box, your output data LFNs are safe
```

5. Cleanup

Following GridStorageQuota.

To see how much space you are using:

You can to go to a new shell and start Dirac:

```
$ SetupProject LHCbDirac
$ lhcb-proxy-init
$ dirac-dms-storage-usage-summary --Dir /lhcb/user/<u>/<username>
```

Or get the same in ganga:

```
In [1]: gridProxy.renew()
In [2]: diracAPI('import commands; result=commands.getoutput(dirac-dms-storage-usage-summary --Di
```

You can remove all copies of files from a given dataset within Ganga:

```
In [1]: ds=j.backend.getOutputDataLFNs()
In [2]: for d in ds
  ....:     d.remove()
```

And finally find out what's left over from Dirac, and exterminate it: You can to go to a new shell and start Dirac:

```
$ SetupProject LHCbDirac
$ dirac-dms-user-lfns
$ dirac-dms-remove-files <a-list-of-lfns>
```

You could also run those commands through the diracAPI in ganga, but since they take a very long time, probably it's not the best idea.

Now it probably takes some time to sync the corresponding DB, containing the storage space used. So please wait a bit.

6. Advanced cleanup

Load the dirac-dms-user-lfns into an LHCbDataset:

```
f=open('<a-list-of-lfns>')
files=f.read().strip().split('\n')
f.close()
for i in range(len(files)):
  while('//') in files[i]:
    files[i]=files[i].replace('//','/')
files=['LFN:'+f for f in files]
ds=LHCbDataset(files)
```

Then go through the datasets you want to keep, subtracting from this list:

```
ds_diff=ds
for ds2 in box:
  ds_diff=ds_diff.difference(ds2)
```

Then remove them:

```
for df in ds_diff:
```

```
df.remove()
```

Tips

Most things can be reduced to a line or two using Ganga Utilities. It's simple to get working and can save you a lot of time.

(1) is just `gu.subjob_resubmit(j)`, to resubmit with changing the settings automatically, and `j=gu.resplit(j)` to make a new job with the failed subjob inputdata

(6) is `ds=gu.dataset_from_file('<a-list-of-lfns>'); keep=gu.boxLFNs(); ds_diff=ds.difference(keep);` to get the list files to remove.

-- RobLambert - 25-Nov-2010

This topic: LHCb > GridAndDataManagement
Topic revision: r13 - 2013-06-04 - RobLambert