Retrieving AGATA data from the GRID

1 Introduction

All AGATA data is transfered after some time from the local hard drives to the so called GRID. The GRID is an international collaboration for storage and processing of data. More explanation can be found in several talks on AGATA weeks.
However the retrieving of data from the GRID is unfortunately not a straightforward task. Problems are a lack of supported systems, documentation in general, real certificates and not well developed software. However it could be done, the only thing that is urgently needed is a certificate from a certificaton authority.

2 The data sets

In general the data sets are transfered to the GRID site with a similar directory layout like during the experiment or replay. The only difference is that many small files are packaged in tar archives. The reason for that is the more effective usage of tape systems.
Data sets from LNL are named `YEAR_WEEK`, from GSI the names are correlated to the spokesperson and the GANIL data with the experiment number (e. g. `e663`).

3 The AGATA virtual organisation

Before you can start you must become a certified member of the AGATA virtual organisiation. There is a small howto on www.agata.org. It states:

> In order to access the AGATA Grid resources, you must be a member of the AGATA Virtual Organization (AVO). This is a short description of how to become a member of the AVO.
>
> You must have a valid Grid Certificate installed on your computer. If you have some active grid users or specialists at your home institute, then ask them for advice on how to get a Grid Certificate. You can also get some advice for this at EGI.eu. When you have installed the Grid Certificate, you can apply for an AVO membership by clicking on this link: AGATA VOMS Server
>
> For further information about the AVO and help to become of member, please contact Johan Nyberg.

In general it is best to ask your local system administrators or web site mantainers from where they get their trusted certificates.

4 Installation

The main supported distributions are the Scientific Linux 5 and 6. In this tutorial the Release 6.7 (Carbon) was installed on a virtual machine. Next one must add the

repositories to download the GRID relevant software. Download the SL6 repo file from `http://emisoft.web.cern.ch/emisoft` and install it via
`yum install emi-release-3.0.0-2.el6.noarch.rpm`
Now all recent packages can be downloaded. The following packages should be installed in order to work with a basic set of tools: `lcg-utils`, `voms-client`, `emi-storm-srm-client-mp`, `lcg-infosites` and `dcache-srmclient`. In addtion some certificate lists should be set up. This is done by adding an extra repository[1]. Here you install the `ca-policy-egi-core` package via yum.
I added to `/etc/grid-security/vomsdir` a folder named `vo.agata.org`. In the folder I added a file with the server name of the AGATA-voms server [2] and the following text (copied from the info site of the AGATA-voms server):

```
O=GRID-FR/C=FR/O=CNRS/OU=CC-IN2P3/CN=cclcgvomsli01.in2p3.fr
/C=FR/O=CNRS/CN=GRID2-FR
```

However still there are some problems using `BDII`. Next your user certificate, user key and user credentials should be copied to the `.globus` folder in your home directory. They should be named `usercert.pem`, `userkey.pem` and `userkey.pem` respectivly.

5 Communication and copy

5.1 voms proxy

In order to register your local machine you have to create a so called voms proxy server. This is done by a command line tool

```
voms-proxy-init -voms vo.agata.org
```

and can be checked by calling `voms-proxy-info` resulting in an output

```
subject   : /C=DE/O=GermanGrid/OU=UniKoeln/CN=Benedikt Birkenbach/CN=proxy
issuer    : /C=DE/O=GermanGrid/OU=UniKoeln/CN=Benedikt Birkenbach
identity  : /C=DE/O=GermanGrid/OU=UniKoeln/CN=Benedikt Birkenbach
type      : proxy
strength  : 1024 bits
path      : /tmp/x509up_u5139
timeleft  : 24:19:30
```

Without this proxy server running you cannot use any grid site.

5.2 Sites

There is a list of available sites at the DAQ AGATA wiki[3]. However you can check the available sites with the command

```
lcg-infosites --vo vo.agata.org ce se
```

[1] https://wiki.egi.eu/wiki/EGI_IGTF_Release

[2] cclcgvomsli01.in2p3.fr.lsc

[3] http://csngwinfo.in2p3.fr/mediawiki/index.php/SRM_StoRM

resulting in the following list:

```
#   CPU    Free Total Jobs Running Waiting ComputingElement
----------------------------------------------------------------
    816     210          0       0       0 ce03.ific.uv.es:8443/cream-pbs-agataL
    816     210          0       0       0 ce03.ific.uv.es:8443/cream-pbs-short
   1364      72          0       0       0 ipngrid04.in2p3.fr:8443/cream-pbs-agata
   1526     234          0       0       0 ipngrid04.in2p3.fr:8443/cream-pbs-sdj
   1712       0          0       0       0 lyogrid07.in2p3.fr:8443/cream-pbs-vo.agata.org
    232     173          0       0       0 sbgce1.in2p3.fr:8443/cream-pbs-vo.agata.org
   1728       0          0       0       0 sbgce2.in2p3.fr:8443/cream-pbs-vo.agata.org
 Avail Space(kB)  Used Space(kB)  Type  SE
------------------------------------------
     21474836480             n.a  SRM   ccsrm02.in2p3.fr
     10690386063      8015407132  SRM   ipnsedpm.in2p3.fr
      1496838844      1003564027  SRM   lyogrid06.in2p3.fr
    138488713288   1039534065768  SRM   sbgse1.in2p3.fr
      4774168844     45225831155  SRM   srmv2.ific.uv.es
    123553419985      8387975348  SRM   storm-fe-archive.cr.cnaf.infn.it
```

There are more options that give different information. For data transfer on the `se` option is important. It is telling you the nodes and the type of the node. So all our nodes are capable of `SRM`.

5.3 Copy

Two basic tools are `lcg-cp` and `srmls`. `srmls` can be used to inspect the contents of the a given remote resource. For example

```
srmls srm://storm-fe-archive.cr.cnaf.infn.it:8444/agata/
```

results (after some time of waiting) in

```
 0 /agata/
      0 /agata/040414_bentley
      0 /agata/060214_installation
      0 /agata/121001_rudolph
      0 /agata/121009_wieland
      0 /agata/121016_gadea
      0 /agata/121113_boutachkov_pietri
      0 /agata/140113_sourcerun
      0 /agata/14N_2H
      0 /agata/150414_post_exp
      0 /agata/190314_pietralla_wieland
      0 /agata/2009_week12
      0 /agata/2009_week22
      0 /agata/2009_week27
      0 /agata/2009_week43
      0 /agata/2009_week46
      0 /agata/2009_week49
      0 /agata/2010_week06
      0 /agata/2010_week07_pietro
      0 /agata/2010_week07
      0 /agata/2010_week09
```

```
    0 /agata/2010_week19
    0 /agata/2010_week21
    0 /agata/2010_week24
    0 /agata/2010_week25
    0 /agata/2010_week29
    0 /agata/2010_week49
    0 /agata/2011_week10
    0 /agata/2011_week11
    0 /agata/2011_week12
    0 /agata/2011_week14
    0 /agata/2011_week19
    0 /agata/2011_week23
    0 /agata/2011_week28
    0 /agata/2011_week29
    0 /agata/2011_week39
    0 /agata/2011_week40
    0 /agata/2011_week41
    0 /agata/2011_week42
    0 /agata/2011_week43
    0 /agata/2011_week44
    0 /agata/2011_week45
    0 /agata/2011_week49
    0 /agata/2011_week50
    0 /agata/2011_week51
    0 /agata/2011_week52_P1
    0 /agata/2011_week52_P2
    0 /agata/2011_week52_P3
    0 /agata/240214_rudolph
    0 /agata/BaseFilesGSI
    0 /agata/caracterisation_Ge
    0 /agata/data2014
    0 /agata/e663
    0 /agata/e664
    0 /agata/e669
    0 /agata/e680
    0 /agata/eagata
    0 /agata/installation_2014
    0 /agata/installation_tests
    0 /agata/performance_commisionning
    0 /agata/psa-tests_01
    0 /agata/psa-tests_02
    0 /agata/psa-tests_03
    0 /agata/psa-tests_04
    0 /agata/psa-tests_05
    0 /agata/psa-tests_06
    0 /agata/test_back
    0 /agata/test_pietro
```

```
230 /agata/test.yann
5368709120 /agata/test_new_ep
560887532 /agata/file001.bin
14632960 /agata/test_small_file_01
36556800 /agata/test_small_file_02
380731392 /agata/test_molini_01
1610383360 /agata/event_mezzdata.0000.test
462 /agata/test_txt_file_01
1347551232 /agata/event_mezzdata.0000.proto
0 /agata/e682
0 /agata/e672
```

The `0` indicates, that the type of the file is a folder. If you now want to copy a file you can use the `lcg-cp` tool:

```
cg-cp -b --vo vo.agata.org -D srmv2 -T srmv2 -v \\
"srm://storm-fe-archive.cr.cnaf.infn.it:8444/ \\
srm/managerv2?SFN=/agata/2011_week41/run_0120/Conf/1B/SRM_AGATA_small_files.tar" \\
"file:/home/bene/Test/SRM_AGATA_small_files.tar"
```

So far so good. We can now copy single files from the GRID to our local system. However if you want to copy a complete experiment to your system you would like to copy recursively the directory and the two tools do not support this. After spending some time looking for tools that do what I want, I decided to program it on my own. The result is a `python` script that uses `srmls` to generate a `bash` script that could be called to download a folder recursively. The gain of flexibility is, that you can generate scripts for each sub run and simultaneously download the files. As most of the time your copy commands will time out, the `bash` script is always waiting for completion. The python script is in the appendix.
In general the time to copy the files is rather long, it might make sense to make sure the voms-proxy is running. The time the proxy is valid can be controlled by the -H parameter.

A Download.py

```
#!/bin/env python

import subprocess
import sys

def usage():
print ""
print ""
print "Download AGATA data from a grid site"
print "srmls (DCache) must be installed"
print "lcg-cp must be installed"
print "More information Documentation section at"
print "http://www.ikp.uni-koeln.de/agata"
print "bene@ikp.uni-koeln.de"
print ""
print "USAGE:"
print "Download.py SRM-Resource SRM-PATH Local-PATH"
print ""
```

```
def rcopy(_grid, _local, _filestring):
gridcopy = "lcg-cp -b --vo vo.agata.org --connect-timeout 20 -D srmv2 -T srmv2 -v "
mkdir = "mkdir -p "
if len(_filestring.split()) != 2:
return
if int(_filestring.split()[0]) != 0:
# OK we have a real file
# the directory should have been generated already
print "while [ ! -e " + _local + _filestring.split()[1] + " ]"
print "do"
print "echo \"Downloading " + _filestring.split()[1] + "\""
print gridcopy + _grid + "/srm/managerv2?SFN=" +  _filestring.split()[1] \\
+ " " + _local + _filestring.split()[1]
print "done"
else:
# So we have a directory
# get all entries from the directory and call yourself
# command = ["lcg-ls", "-v", "-b", "-D" "srmv2", _grid + _filestring.split()[1]]
command = ["srmls", _grid + _filestring.split()[1]]
print "echo \"Making directory " + _local + _filestring.split()[1] + "\""
print mkdir + _local + _filestring.split()[1]
result = subprocess.Popen(command, stdout=subprocess.PIPE).communicate()[0]
for filestring in result.split("\n"):
if len(filestring.split()) == 2:
# DEBUG print "######################################################################"
# DEBUG print " CALLED:                              CURRENT:  "
# DEBUG print _filestring + "          " + filestring
# DEBUG print _filestring.split()[1] + "       " +  filestring.split()[1]
# DEBUG print "----------------------------------------------------------------------"
if filestring.split()[1] != _filestring.split()[1]:
rcopy(_grid, _local, filestring)
if len(sys.argv) != 4:
usage()
sys.exit(1);

print "#!/bin/bash"
print "echo \"Trying to recursively copy" + sys.argv[1] + "\""
print "echo \"to the local path " + sys.argv[2] + "\""

grid = sys.argv[1]
filestring = "0 " + sys.argv[2]
local = sys.argv[3]

rcopy(grid, local, filestring)
```

The script is called from the command line:

```
Download.py srm://storm-fe-archive.cr.cnaf.infn.it:8444 /agata/2011_week42/ /local/data/ > dl.sh
```