Exercises for Grid Application Toolkit

Part I: Getting started

NOTE: Replace XX with your team number.

The Grid Application Toolkit is installed in /opt/gat/. The adaptors are located at /opt/gat/lib/GAT/adaptors. In order to set the environment variables logon to gridlab1.phys.utb.edu do the following:

$ **source /opt/gat/gatenv.sh**

This will export two environment variables namely the GAT_LOCATION that points to the folder where the Grid Application Toolkit engine is installed. The second GAT_ADAPTOR_PATH tells the engine the location of the adaptors to be loaded when the engine loads up.

$ **echo $GAT_LOCATION**
/opt/gat
$ **echo $GAT_ADAPTOR_PATH**
/opt/gat/lib/GAT/adaptor-list-c

The GAT_ADAPTOR_PATH points to a text file that contains the names of adaptor libraries.
$ **cat $GAT_ADAPTOR_PATH**
In this exercise we will write GAT programs to move files between resources using the GridFTP adaptor that uses the gsiftp protocol to transfer files and to submit jobs using the GAT. In order to test file transfer we will as before create files:
$ **dd if=/dev/zero of=./testfileYOURNAME bs=1M count=10**
Again to check the size
$ **ls -lh ./testfileYOURNAME**
PART II: GAT Program to transfer files
Make a directory that will house your gat programs:
$ **mkdir gatprg**
$ **cd gatprg**
Now lets type the C Program that will copy files locally and between machines!!!!
$ **cat > copyfile.c**

```
/* System Header Files */
#include <stdio.h>

/* GAT Header Files */
#include "GAT.h"
#include "GATTestUtils.h"

int main (int argc, char *argv[])
{
  GATContext  context = NULL;
  GATLocation name1   = NULL;
  GATLocation name2   = NULL;
```

```
  GATFile     file1   = NULL;
  GATResult   retval  = GAT_FAIL;

  /* check for correct invocation */
  if ( argc < 2 )
  {
    printf ("\n\tUsage: %s <src> <target>\n"
          "\n\tprogram does:\n"
          "\n\t\tcp <name1> <name2>\n\n" , argv[0]);
    exit   (1);
  }

  /* initialize GAT: create context */
  context  = GATContext_Create   ();

  /* create URLs for all file names */
  name1    = GATLocation_Create  (argv[1]);
  name2    = GATLocation_Create  (argv[2]);

  /* create initial file object */
  file1    = GATFile_Create (context, name1, 0);

  /* cp <name1> <name2> */
  retval = GATFile_Copy   (file1, name2, GATFileMode_Overwrite);
  GAT_TEST_TRACE(retval == GAT_SUCCESS, context);

  /* clean up */
  GATFile_Destroy     (&file1);
  GATLocation_Destroy (&name1);
  GATLocation_Destroy (&name2);
  GATContext_Destroy  (&context);

  return (0);
}
 Ctrl-D
```

Test if your file is there…

$ **cat copyfile.c**

Next we need to compile our GAT Program. In order to do this we will need a Makefile that will ease the job of compiling our GAT programs. We will also need to do a grid-proxy-init to be able to use gisftp to transfer files.

$ **cp /home/architk/gatprg/Makefile .**

$ **make**

Now generate a proxy that we will use to transfer files and submit jobs using GridFTP and GRAM.

$ **grid-proxy-init**

Now you will see an executable file with the name of the C source file (without the extension) that you
created
$**./copyfile /home/trainXX/testfileYOURNAME /home/trainXX/testfile**
Check if your file exists.

Transferring files to remote systems. Now we will switch the adaptor used by GAT and use the same program to transfer files
The environment variable GAT_ADAPTOR_PATH points to a file containing the list of adaptors available to GAT. These can be switched by modifying this file.
We will create a new adaptor-list with the gridftp adaptor to use gridftp to transfer files.

$ **cat > adaptor-list-globus**
/opt/gat/lib/GAT/adaptors/libgridftp_adaptor.so
/opt/gat/lib/GAT/adaptors/libgram_adaptor.so
/opt/gat/lib/GAT/adaptors/libfileops_adaptor.so
/opt/gat/lib/GAT/adaptors/liblogicalfile_adaptor.so
/opt/gat/lib/GAT/adaptors/libresourcebroker_adaptor.so
/opt/gat/lib/GAT/adaptors/libservicebroker_adaptor.so
/opt/gat/lib/GAT/adaptors/libfilestream_adaptor.so
/opt/gat/lib/GAT/adaptors/libendpoint_adaptor.so
/opt/gat/lib/GAT/adaptors/libadvertservice_adaptor.so
Ctrl-D

$ **export GAT_ADAPTOR_PATH=/home/trainXX/gatprg/adaptor-list-globus**

Now execute:

./copyfile gsiftp://gridlab1.phys.utb.edu/home/trainXX/testfileYOURNAME gsiftp://gridlab2.phys.utb.edu/home/trainXX/testfile

Logon to gridlab2 to check if the file exists.

PART III: Submitting Jobs
Now we will write a GAT Program that reads a command line argument that is the name of an executable and some arguments and executes it on the local system.
Type out/paste the following program. The program can be found at the Syllabus page. To get it from there use:

$ **wget**
http://osg.ivdgl.org/twiki/pub/SummerGridWorkshop/SummerGridSyllabus2006/submitjob.c

In the file find the line that reads

GATTable_Add_String(attributes, "stdout", "/home/train36/result.out");
and change train36 to your userid (trainXX).

Now as before we will compile and link the C program using the make command.
To do this simply
type the make command that uses the Makefile that we wrote earlier.

$ **make**

This will generate the executable file submitjob.

$**./submitjob /bin/date**

This will submit /bin/date to the local system.

Now we will submit a job to a remote machine via GRAM.

We will replace the local resourcebroker adaptor with the GRAM adaptor:

Use an editor to edit the **/home/trainXX/gatprg/adaptor-list-globus** to look like:
Note that the # in front of the gram adaptor entry is deleted to enable the GRAM adaptor.

```
 /opt/gat/lib/GAT/adaptors/libgridftp_adaptor.so
 /opt/gat/lib/GAT/adaptors/libgram_adaptor.so
 /opt/gat/lib/GAT/adaptors/libfileops_adaptor.so
 /opt/gat/lib/GAT/adaptors/liblogicalfile_adaptor.so
 /opt/gat/lib/GAT/adaptors/libresourcebroker_adaptor.so
 /opt/gat/lib/GAT/adaptors/libservicebroker_adaptor.so
 /opt/gat/lib/GAT/adaptors/libfilestream_adaptor.so
 /opt/gat/lib/GAT/adaptors/libendpoint_adaptor.so
 /opt/gat/lib/GAT/adaptors/libadvertservice_adaptor.so
# /opt/gat/lib/GAT/adaptors/libtracing_adaptor.so
```

Now create a file that will hold a list of remote resources that you can submit to:

$**cat > resources.ini**

```
[resources.gridlab2-globus]
 memory.size          = 0.256000
 memory.accesstime    = 0.25
 memory.str           = 1
 machine.type         = unknown
 machine.node         = gridlab2.phys.utb.edu
 machine.nodecount    = 1
 machine.address      = gridlab2.phys.utb.edu
 machine.maxpos       = 2
```

```
  machine.minpos       = 1
  rms.name             = none
  rms.host             = gridlab2.phys.utb.edu
  rms.version          = 10.000000
  rms.queues           = none
  rms.availableprocs   = 2
  rms.load             = 45
  rms.maxtime          = 0:45
  rms.maxwalltime      = 4:00
  cpu.type             = unknown
  cpu.speed            = 3.000000
  cpu.count            = 1
  vm.count             = 4
  disk.size            = 10.000000
  disk.virtualsize     = 1000
  disk.filesystem      = ext3
  disk.accesstime      = 0.01
  disk.str             = 1
  bandwidth            = 130
  latency              = 0.1
  interconnect.type    = none
  ssh_type             = globus
  ssh_port             = 2222
Ctrl-D
```

Now do

$**export GAT_RESOURCE_INI=/home/trainXX/gatprg/resources.ini**

Now run the same submit job program again:

$./submitjob /bin/date

Log on to gridlab2 and look for result.out in your home directory.
You should see todays date and current time in the file.

EXTRA CREDIT: Write a GAT Program that transfers a binary (write a small program if needed) to a remote system and then submits it to the machine.