AHE Practical 2: Scripting AHE Client Workflows

Aims and Objectives

- Automate common application workflows by scripting the AHE command line clients
- Configure the AHE command line clients to help automate scripting
- Understand the concepts behind AHE client workflows
- Modify the example scripts to perform more complicated tasks

Introduction

By calling the AHE clients from scripts, complicated application workflows can be achieved. AHE workflows follow the pattern of running the ahe-prepare command followed by the ahe-start command, followed by other commands needed to carry out the workflow (such as ahe-monitor or ahe-getoutput).

Any scripting language can be used to call the AHE clients and perform the workflows. In the following example we have used Perl – a detailed knowledge of Perl isn't necessary to run the scripts; the comments in the code explain the actions taking place.

Stage 1: Preparing to run workflows

1 The workflows in this exercise make use of the ahe-client installation from exercise 1. Ensure that the **$AHECLIENT_HOME** variable is set to the location of your AHE client installation.

2 The AHE command line tools make use of a local cache of jobs to improve performance. Before running the workflow scripts it is necessary to configure the location of the client cache. To do so edit the **aheclient.properties** file in the $AHECLIENT_HOME/conf folder. For example:

```
emacs $AHECLIENT_HOME/conf/aheclient.properties
```

The cache file can be saved into /tmp. Change the line:

```
uk.ac.ucl.chem.ccs.aheclient.cache=.ahecache
```

to

```
uk.ac.ucl.chem.ccs.aheclient.cache=/tmp/ahecache
```

3 When running the AHE command line clients, by default you will be asked for your keystore password. When automating tasks by scripting clients it is necessary to configure the password in the aheclient.properties file, so that the command line clients can be called without user input.

To do so add your password to the right-hand side of the = sign of the following line:

```
uk.ac.ucl.chem.ccs.aheclient.passwd=
```

4 Before running the command line clients, ensure that you have generated and uploaded a proxy certificate using the MyProxy upload tool. You will need to enter your username and password in the ahe configuration file:

```
uk.ac.ucl.chem.ccs.aheclient.myproxy-pw=<your_password>
uk.ac.ucl.chem.ccs.aheclient.myproxy-un=<your_username>
```

Stage 2: Automating job submission

1 Usually when launching an AHE job from the command line, the user has to issue the ahe-prepare command to create a web service to manage the application run, followed by the ahe-start command to start the job. In this part of the exercise we will create a script which will automate the prepare and start commands.

2 Copy the autojob.pl script to your home directory:

```
cp /opt/NGSAppDev/AHE/ex2/autojob.pl ~/
```

The script is shown below. The comments in the code describe the tasks taking place.

The script automates the task of preparing a sort application job and executing it on the NGS Manchester node.

```
#!/usr/bin/perl -w

use strict;
use warnings;

# set the confFile variable to the first argument of the script
my $confFile = $ARGV[0];

# create a unique job name based on the time
my $time = time();
my $jobName = 'testjob' . $time;

# set the prepare_command variable to the command that you would
execute from
# the terminal, setting the job name to the unique name generated
above

my $prepare_command = "$ENV{AHECLIENT_HOME}/bin/ahe-prepare -s
$jobName -app sort -e
https://chemd237.chem.ucl.ac.uk:9443/ahe/AppWSResource";

# print out the prepare command
print $prepare_command . "\n\n";

# execute the prepare command
system $prepare_command;

# set the start_command variable to the command that you would
execute from
```

```
# the terminal, setting the job name to the unique name generated
above
# and the configuration file specified as the argument to the
script
# the job will be run on the Manchester NGS node

my $start_command = "$ENV{AHECLIENT_HOME}/bin/ahe-start -s
$jobName -config $confFile -RM Manchester -n 1 -wallTimeLimit 1";

# print the start command
print $start_command . "\n\n";

# execute the start command
system $start_command;
```

3 Change to the directory containing the autojob.pl script. Execute the script using the sort application input files downloaded in practical 1 as follows:

./autojob.pl $HOME/sortapp-input/config.txt

4 Use the commands ahe-monitor and ahe-getoutput to monitor the job once submitted.

Stage 3: Running multiple jobs

1 Access to the wide variety of resources that the grid provides makes it possible to automate the submission of a large number of jobs. Expanding on the script created in stage 2, we will automate the submission of jobs to several different machines on the NGS.

In this example we will run the same sort job on machines at Manchester and RAL. The script uses a loop to run the ahe-prepare and ahe-start commands twice, iterating through an array of NGS machines and submitting to them in turn.

2 Copy the multijobs.pl script to your home directory:

cp /opt/NGSAppDev/AHE/ex2/multijobs.pl ~/

The script is shown below. The comments in the code describe the tasks taking place.

```
#!/usr/bin/perl -w

use strict;
use warnings;

# set the confFile variable to the first argument of the script
my $confFile = $ARGV[0];

# set the number of jobs to perform
my $numJobs = 2;

# set an array containing the names of the NGS nodes to use
# you can find the names of NGS machines by running the AHE
prepare command
# manually - in this case we are submitting to Manchester and RAL
```

```
my @RMArray = ("Manchester", "RAL");

# create a unique job name used as a base for all simulations
lauched by
# this script
my $time = time();
my $jobName = 'multijob' . $time;

for(my $i = 0;$i < $numJobs; $i++){

    print "Performing job " . $i;

# set the array index
    my $arrayIdx = $i % scalar(@RMArray);

# set the individual job name from the base job name and loop
iteration
    my $individualJobName = $jobName.$i;

# set the prepare_command variable to the command that you would
execute from
# the terminal, setting the individual job name to the unique
name
# generated above

    my $prepare_command = "$ENV{AHECLIENT_HOME}/bin/ahe-prepare -
s $individualJobName -app sort -e
https://chemd237.chem.ucl.ac.uk:9443/ahe/AppWSResource";

# print out the prepare command
    print $prepare_command . "\n\n";

# execute the prepare command
    system $prepare_command;

# set the start_command variable to the command that you would
execute from
# the terminal, setting the individual job name to the unique
name generated
# above and the configuration file specified as the argument to
the script

    my $start_command = "$ENV{AHECLIENT_HOME}/bin/ahe-start -s
$individualJobName -config $confFile -RM $RMArray[$arrayIdx] -n 1
-wallTimeLimit 1";

# print the start command
    print $start_command . "\n\n";

# execute the start command
    system $start_command;

}
```

3 Change to the directory containing the multijobs.pl script. Execute the script using the sort application input files downloaded in practical 1 as follows:

./multijobs.pl $HOME/sortapp-input/config.txt

4 Use the command line clients to monitor both of the jobs launched by the script.

Stage 4: Stacking jobs end on end

1 Grid resource usage policies often dictate a maximum amount of time that a job is run for (the walltime limit). It can be necessary to split a long running job into shorter sections, which will run within the time limit.

This script will stack sort jobs end on end, launching one on the NGS Oxford machine and waiting for it to finish before starting the next. The input parameters to the script are the config files of the different sort jobs to launch.

After preparing and starting a job, the script sits in a loop using the ahe-monitor command to poll the job status every minute. The output of the monitor command is saved in a variable – when the status change to Job complete the loop breaks and the next application is launched.

2 Copy the stackedjobs.pl script to your home directory:

cp /opt/NGSAppDev/AHE/ex2/stackedjobs.pl ~/

The script is shown below. The comments in the code describe the tasks taking place.

```
#!/usr/bin/perl -w

use strict;
use warnings;

# set the number of jobs to the number of input files specified
my $numJobs = $#ARGV + 1;

# create a unique job name used as a base for all simulations
lauched by
# this script
my $time = time();
my $jobName = 'stackedjob' . $time;

for(my $i = 0;$i < $numJobs; $i++){

    print "Performing job $i \n";

# set the individual job name from the base job name and loop
iteration
    my $individualJobName = $jobName.$i;

# set the prepare_command variable to the command that you would
execute from
# the terminal, setting the individual job name to the unique
name
# generated above

    my $prepare_command = "$ENV{AHECLIENT_HOME}/bin/ahe-prepare -
s $individualJobName -app sort -e
https://chemd237.chem.ucl.ac.uk:9443/ahe/AppWSResource";

# print out the prepare command
    print $prepare_command . "\n\n";

# execute the prepare command
    system $prepare_command;

# set the start_command variable to the command that you would
execute from
# the terminal, setting the individual job name to the unique
name generated
# above and the configuration file specified as the argument to
the script
# all jobs sill be sent to the Manchester NGS machine
# the arguments will be iterated through to set the config file
to use

    my $start_command = "$ENV{AHECLIENT_HOME}/bin/ahe-start -s
$individualJobName -config $ARGV[$i] -RM Manchester -n 1 -
wallTimeLimit 1";

# print the start command
    print $start_command . "\n\n";

# execute the start command
system $start_command;

# set variable to hold status of job
    my $jobStatus = "";

# loop and perform monitoring command until status of job is Job
complete
    while(!($jobStatus =~ /Job completed/)){

        print "Polling $individualJobName\n";

# wait for 1 min before polling
        sleep(60);

#use ahe_monitor command to poll job state.
        my $monitor_command = "$ENV{AHECLIENT_HOME}/bin/ahe-
monitor -s $individualJobName";
        print $monitor_command . "\n\n";

#execute the monitor command, and store the output in jobStatus
        $jobStatus = `$monitor_command`;
    }

    print "Job $i complete\n";

}
```

3 To run the script copy the sortapp-stacked.tgz input file set to your home directory:

cp /opt/NGSAppDev/AHE/ex2/sortapp-stacked.tgz ~/

Unpack the file set with:

```
tar zxvf sortapp-stacked.tgz
```

Assuming in the input files have been downloaded to the desktop, change to the directory containing your stackedjob.pl script and run the script as follows:

```
./stackedjob.pl $HOME/sortapp-stacked/config1.txt $HOME/sortapp-stacked/config2.txt $HOME/sortapp-stacked/config3.txt
```

The above command should all be on the same line.

4 Again, check the progress of the stacked jobs with the command line clients.

Further Work

- Extend the multjobs.pl script to submit more than two jobs
- Modify the mulitjobs.pl script to submit different jobs using the config1.txt – config3.txt files from the sortapp-stacked input file set
- Modify the stackedjob.pl script to download the each job’s output files once it is complete.
- Modify the stackedjob.pl script to check for failed jobs.