

---

# Distributed Computing for eScience 2

Group project : grid middleware

## Group report

Rajesh Babu  
Albert Heyrovsky  
Ma Xin Hong  
Yating Li  
Mark MacGillivray  
Tom Omitowoju  
Steven Truong

---

---

**Contents**

Abstract ..... 3

Introduction ..... 3

Requirements ..... 4

Project Tasks ..... 6

Initial Testing ..... 7

Development Issues ..... 8

Evaluation ..... 11

Appendix ..... 13

**Abstract:** An analysis of a parameter sweep type task in a bid to search for knowledge. The tools used to perform the parameter sweep were gLite, Globus Toolkit (GT4) and Condor, three popular middleware in Grid technology. The report continues with an analysis on the functionality of these tools as a system in a bid to decrypt hidden messages on a large scale.

## **Introduction**

The ad-hoc group consists of seven members with similar background knowledge in the e-Science field. The group have been told that in some distant land there are hints of alien activity. The aliens have left floating box-like objects called pillars that exist in some 2D-space. Atop each pillar exists a message, there is a belief that the aliens have left a secret message to whomever has the technology to find it.

There are 3 known spaces that are populated with pillars, they exist in the Globus, Gilda and Condor spaces, the spaces have already been dumped on to their respective nodes. Currently, little is known on how many pillars exist or where they are located. Another problem discovered is that the spaces are not flat, but have wavy noises everywhere i.e. hills or ditches. A method to zoom in and out of the space is required to read the pillars from the noisy space accurately.

## Requirements

### Functional

The functional requirements are applicable to most e-Science problems that involve a parameter sweep to search for knowledge - these typically are:

- A tool to search through the space, in this project having one tool that works in all spaces is a distinct advantage.
- Middleware clients that run on the three spaces to allow the submission of jobs with minimal worry about the heterogeneity of the different tools and layers of Grid computing.
- The spaces are large, having a tool that informs the user of a job completion can maximise the team's overall efficiency.
- A data repository to store the messages.

Fortunately there are already existing technologies to handle each functional requirement. The tool to scan through the space has already been installed on the nodes containing the relevant spaces. The scanner tool was developed in Java and therefore it is easily portable and reusable on different systems.

The three middleware used to submit the scanner jobs are well renowned in Grid computing. The three used are; Globus Toolkit 4 (GT4) on the Globus space, Condor on the Condor space and gLite on the Gilda space. Using these 3 pieces of middleware the team are able to submit the scanner jobs on each space to discover the respective pillars.

The data repository is required so the team can submit their discoveries in a location that is accessible to each member. OGSA-DAI is a middleware product that allows a user to access data resources via web services. The benefit of using OGSA-DAI is that it offers a flexible approach to data retrieval in a Grid environment. The data repository can be in the form of a variety of standards such as relational or XML databases.

### Non-Functional

In e-Science type projects, teams are built up from different parts of the world. The team cannot be expected to travel such distances for meetings or to work on the project, this defeats the concepts behind Grid as being a global infrastructure.

There are several steps of communication that are dependent on the stage of the project. The first step is to establish communication as a group. The first step was to obtain each member's contact details, typically their e-mail address. The e-mails are then combined into an e-mail list for simplicity. The can then be used to decide when the first meeting will be held.

As mentioned previously, members are located across large distances and cannot be expected to travel for regular meetings. The team used Skype to hold the meetings. Skype is a free client that allows people to make internet phone calls at no cost. It is a relatively small client and is fast to install and easy to register with.

The team might be in different time zones and their work on the project might be stalled until another person completed some part of code. Therefore a mechanism is required to inform the team members of progress in a timely manner. Twitter is a social networking tool that is capable of sending SMS messages on request to keep people up to date with each other. It also offers an API to allow programmers to update their Twitter status. For example, once a job is discovered it is published in the OGSA-DAI repository using some script (or code) the script can be extended to also update the user's Twitter status, Twitter will then send out a notification to "Twitter followers" of that user.

## **Project Tasks**

There are particular tasks that must be completed within the time available in order to achieve the project goal. These tasks must also take account of the project requirements and constraints, such as understanding the Globus, Gilda, and Condor middlewares.

As it is not known how many pillars are in each space, it is necessary to run tests on all the available clusters under each middleware. There are 3 grid systems, and each of them contains some clusters, thus, a team member is designated as responsible for each middleware. Also, a team member should be responsible for testing, and another for keeping records. That is not to say that a team member will work only within their responsibility, but simply to allocate someone to ensure each task area is proceeding sensibly. The final team member can work between groups as required, and ensure co-ordinated working of the groups.

An example co-ordinate has been provided for each cluster; using this the time taken for running particular jobs on each cluster can be recorded. A test is also necessary for having multiple job submitters, in case problems occur, such as running out of memory. In order to solve the problem, one solution may be to shrink the size of the job, second solution is to increase the step size and decrease the range of the coordinate for searching.

The next consideration is the time; by considering the amount of work the group has time for, and using the three point estimate to get a rough idea on how many days need to be taken for finishing the project. After planning it is estimated that the practical work will take seven days to complete. This leaves time for the last part, which is to review outcomes achieved and data retrieved; if successful, having all the results from the three different systems, the results can be written up. Otherwise, extra time can be used to solve remaining issues.

As the project runs, it will also be necessary to have regular updates as the team will be working in a distributed fashion for much of the time. Meeting records, and records of issues discovered or of results found, will be kept. This will also provide the required content for the report. In order to ensure the project keeps running, a weekly development meeting will be scheduled. Also for updates, a weekly Skype meeting can be held. Twitter can be used to send group messages, either by individuals or automated within job sessions, and team members can communicate issues and discoveries by email too. Finally, a wiki will be kept for providing any development or other notes that may need to be referenced later.

## Initial Testing

Testing is performed on the various middlewares, to get a better understanding of how the scanner provides results, and of how it runs on the different systems. When the step size is incorrect, the ascii art word is only partially displayed. Thus the scans will need to be run at iterations with different step size to get the complete answers. Ideally, the process of checking results would be automated, and enable starting of new jobs at new step sizes, but there may not be time to develop such automation. So jobs must run fast enough to enable manual checking at this stage, if required.

Twitter and OGSA\_DAI APIs allow messaging when results sets are found, and updating of the database with result information. This works successfully. Now, it is necessary to define what search sizes should be run.

Varying grid and step sizes were tested for a test job run directly on Condor platform, also with differing memory settings :

X and Y	Step size	Memory allocated (GB)	Time taken (seconds)
<b>200</b>	0.1	1	3
<b>400</b>	0.1	1	9
<b>600</b>	0.1	1	20
<b>800</b>	0.1	1	Out of memory
<b>200</b>	0.06	1	6.7
<b>600</b>	0.06	1	Out of memory
<b>800</b>	0.1	2	35
<b>1000</b>	0.1	2	Out of memory
<b>1000</b>	0.1	3	54
<b>1000</b>	0.1	4	Excess heap size

The differing JVMs and machines on the clusters may not support higher than standard memory amounts. Some machines are discovered to be 64 bit, some have 2GB memory (e.g. mc01), others have more. Therefore the team chose to go with standard 1GB memory, using an initial grid size of 250 by 250, with step size 0.1. This should run quickly per job, and should not cause memory errors on the most standard machines. Also, given the previous result estimates, and in order to save processing time, the initial search space is limited to -10000 to 2000 x, -2000 to 10000 y. If all pillars are not discovered, outlying areas can be searched subsequently.

## **Development Issues**

There were various issues discovered during the course of development.

### Initial Setup

When work started on the available systems, beginning with exploratory testing, some systems were immediately available whilst others were not.

There were no initial issues with Condor.

Globus did not work to begin with, we had to get some settings changed by the administrator.

gLite also did not work at the beginning, and we found ongoing issues with it. For some reason, once on the Gilda clusters we were able to run test jobs on the main Gilda.tutor machine and they would complete within seconds. But submitting the same job to be distributed to the Gilda clusters resulted in the jobs taking approximately 30 minutes to complete. We are unsure if this was a result of gLite issues, or just down to the administration and performance of the Gilda clusters.

### Secure Access

Grid middleware systems employ security certificate mechanisms to control authentication and authorisation to the resources that grid services provide. Whilst each team member is used to using and in most cases had already obtained the necessary certificates, there were obstacles to successfully gaining access.

In the case of Condor, access was not an issue. Each team member was able to use the Condor system with their own certificates. Condor was up and running from the beginning of the project.

Globus, on the other hand, presented some small issues. Whilst it seemed that access was straightforward, there were anomalies in use. Every team member had working certificates and was able to successfully create a grid proxy, but in two cases the required delegation of the proxy failed when attempting to submit jobs. Firstly, Albert discovered this issue, but it was solved by requesting a change to the gridmap file on the globus administration system. However, when the same issue was encountered by Mark, altering the gridmap file did not help. Globus access therefore provided an outstanding issue in one case. However, as there were other team members with working access, it was not a serious issue.

For the gLite system, quite a few access problems were encountered, although these were more a result of the Gilda access methods than gLite middleware itself. The team found that obtaining usable certificates from Gilda was achievable but not reliable. Some team members received their initial short

term certificates and were also able to upgrade to one year certificates without issue. Other team members found that their certificates expired despite a request for continuation, or that their attempts to re-use newly acquired logins with old certificates caused confusion or failures. However, given that Gilda is a testbed platform that provides no guarantees on usability, these issues are hardly unforgivable. Overall Gilda still provides a useful service for those wishing to practice with grid middleware.

### Job Submission

There are many jobs to submit, requiring different submission mechanisms. In order to minimise the workload for this, the team was able to develop bash scripts to create jdl or rsl files as required, and further scripts to submit the created jdl/rsl files. This enabled automation of job submission to a certain extent, based on results from previous iterations.

### Memory Limitation

The team found during exploratory testing that the available amounts of memory to run jobs was, in some cases, limited. As the scanner tool is a java program, there is a limit both on the amount of memory available to the virtual machine instantiated when a job is run, as well as the limit of physical memory actually available to a cluster machine assigned the task of running a job. Thus the area that could be searched during any given job had to be limited to ensure jobs did not fail due to running out of memory.

### Performance

Condor, which was the easiest middleware to use from the outset, had few issues whilst running jobs. Machines on the clusters appeared to run efficiently, and most jobs completed quickly. There were some job failures on Condor, but they did not take long to run again. If jobs were submitted to Condor at a rate higher than one per 10 seconds, they would start building up in an idle queue. Having multiple job submitters also resulted in longer queues, thus did not benefit overall completion rates.

Due to the proxy delegation problems, Globus jobs were initially delayed. However once the jobs were started, they ran very quickly with no errors or failures. Using RSL files made it very easy to submit all jobs and leave them to run. Therefore Globus turned out to be the most efficient overall. Globus had no issues accepting job submissions every 5 seconds or more.

gLite had issues from the beginning and continued to do so, although many of these were down to the Gilda platform rather than gLite itself. Some clusters on Gilda appeared to be unresponsive, whilst others had restrictive performance capability. Jobs on gLite took the longest to complete, although they were successfully run in parallel, so the lengthy time per job was not such a drastic issue as it would otherwise have been. Using gLite on Gilda required significantly more customisation of the JDL files than the other platforms. There were also the most number of job failures on Gilda, requiring resubmission to

complete. Individual jobs on Gilda were taking up to 30 minutes to run, resulting in job batches taking longer than the length of the proxy invoked on them. This also resulted in job failures when the proxy ran out after 12 hours. In order to compensate for job failures, a retry statement was added to the jdl files. Also, running on short Gilda queues resulted in better response time than the long or infinite queues, where very little processor time seemed to be available (most likely as they are for longer jobs).

## Evaluation

By the end of the project, all the pillars had been discovered :

Grid	X 1	Y 1	X 2	Y 2	Step size	Plaque word
gLite	-4816.30	6771.20	-4750.10	6790.00	0.05	NeSC
	-5107.40	1754.30	-5079.50	1761.10	0.015	eScience
	-5657.10	6831.00	-5641.20	6834.80	0.01	Condor
	-5682.10	308.10	-5642.20	317.90	0.025	Toolkit
	-4219.60	3458.60	-4155.70	3474.40	0.035	Distributed
	-2416.60	3009.30	-2364.70	3022.10	0.035	Globus
	-2969.90	7723.20	-2966	7724	0.003	WSRF
Globus	-8712	63.6	-8640.1	81.4	0.05	EPCC
	-8348.70	7794.50	-8295.80	7806.30	0.025	Edinburgh
	-6928	3112.20	-6904.10	3118	0.01	Middleware
	-6872.60	5679.30	-6812.70	5694.10	0.03	Internet
	-6052.70	1312.30	-6016.80	1321.10	0.025	OGF
	-9714.20	4765.30	-9702.30	4768.10	0.01	Linux
	Condor	-489.20	5520.60	-481.30	5522.40	0.005
-162.40		8907	-94.50	8923.80	0.05	MSc
53.30		4646.70	73.20	4651.50	0.01	OGSA-DAI
700.30		4884.40	-668.40	4892.20	0.02	Twitter
1267.70		7565.50	1323.60	7579.30	0.03	Resource
1667.30		4717	1711.20	4727.80	0.03	HPC

Discounting the provided example pillars (EPCC, NeSC and HPC), the initials of each discovered word spell out :

### Welcome to the grid

Whilst the project goals were achieved the team discovered:

- out of the three middleware products -  
 Condor is initially easiest to set up and use  
 Globus runs best and therefore becomes easiest over time  
 gLite (on Gilda) is more complicated and less reliable throughout  
  
 Globus gave no errors and dropped no jobs  
 Condor dropped some jobs  
 gLite dropped the most jobs
- Gilda access issues remained unresolved for the duration of the project

- Some Globus proxy delegation issues remain
- Twitter is useful for quick and automated messaging, but is not a convenient method for recording discussion history, like email
- Skype is a useful tool, though it requires some setup and particularly good network connection speeds for each user, otherwise calls are dropped and conference calling can be difficult
- There were no issues using the OGSA-DAI database, once the team got used to the API
- The team was able to work well across a distributed environment
- Overall the work was completed successfully, although it would be good to develop further automation of tasks
- Whilst failing jobs were a small issue, this was mostly on the Gilda service which is a test service providing no service level guarantees

In conclusion, the team found all the pillars in every area, showing that each middleware solution is capable of handling the job; also, it is clear that Globus and Condor were faster and more reliable in this instance than gLite on Gilda; finally, Condor is best to work with in the short term but Globus wins out in the end. Further use of OGSA-DAI and Twitter APIs allows good interconnection between jobs, and improves the ability of the group to work in a distributed manner. These methods, combined with email, voip and wiki communication enable a group to work on a task from different locations as necessary.

## Appendix

### Figure 1: A script for making rsl files

Converts an rsl template (fig.4) into a real rsl for submission

---

```
#!/bin/bash
sixth=1
for ((i=-10000;i<2000;i+=250,ith+=1)); do
  for ((j=-2000;j<10000;j+=250,sixth+=1)); do
    first=$i
    second=$j
    third=`expr $i + 250`
    fourth=`expr $j + 250`
    sed -e 's/#1/"$first"/g
          s/#2/"$second"/g
          s/#3/"$third"/g
          s/#4/"$fourth"/g
          s/#5/0.1/g
          s/#6/"$sixth"/g' template.rsl >> rsl/globus$sixth.rsl
  done
done
```

---

### Figure 2: A script for making jdl files

Converts a jdl template (fig.5) into a real jdl for submission

---

```
#!/bin/bash
sixth=1
for ((i=-10000;i<2000;i+=250,ith+=1)); do
  for ((j=-2000;j<10000;j+=250,sixth+=1)); do
    first=$i
    second=$j
    third=`expr $i + 250`
    fourth=`expr $j + 250`
    sed -e 's/#1/"$first"/g
          s/#2/"$second"/g
          s/#3/"$third"/g
          s/#4/"$fourth"/g
          s/#5/0.1/g
          s/#6/"$sixth"/g' template.jdl >> jdl/condor$sixth.jdl
  done
done
```

---

Figure 3: An example gLite jdl

A gLite jdl ready for submission to run jobs

---

```
Type = "Job";
JobType = "Normal";
Executable = "/usr/bin/java";
Arguments = "-jar sfkscanner.jar gLite 1750 9750 2000 10000 0.1";
StdOutput = "gLite-pillar.out.2304";
StdError = "gLite-pillar.error.2304";
InputSandbox = {"sfkscanner.jar"};
OutputSandbox = {"gLite-pillar.out.2304", "gLite-pillar.error.2304"};
Requirements = ( (!RegExp("-infinite$", other.GlueCEUniqueid)) &&
                (!RegExp("-long$", other.GlueCEUniqueid)) &&
                (other.GlueCEUniqueid !=
"vega-ce.ct.infn.it:2119/jobmanager-lcgsge-gilda") &&
                (other.GlueCEUniqueid !=
"gn0.hpcc.szta.ki.hu:2119/jobmanager-lcgpbs-gilda") &&
                (other.GlueCEUniqueid !=
"ce-edu.grid.acad.bg:2119/jobmanager-pbs-gilda") &&
                (other.GlueHostMainMemoryRAMSize >= 2048) );
Rank = other.GlueCEStateFreeCPUs;
RetryCount = 2;
ShallowRetryCount = 3;
```

---

Figure 4: An rsl template

A template rsl file, to be updated by the rsl script (fig.1)

---

```
<job>
<executable>/usr/bin/java</executable>
<directory>${GLOBUS_LOCATION}</directory>
<argument>-jar</argument>
<argument>sfkscanner.jar</argument>
<argument>Globus</argument>
<argument>#1</argument>
<argument>#2</argument>
<argument>#3</argument>
<argument>#4</argument>
<argument>#5</argument>
<stdout>/tmp/stdout.#6</stdout>
<stderr>/tmp/stderr.#6</stderr>
</job>
```

---

### Figure 5: A jdl template

A template jdl file, to be updated by the jdl script (fig.2)

---

```
Universe = java
Executable = ../sfkscanner.jar
jar_files = ../sfkscanner.jar

Output = condor-pillar.out.#6
Arguments = uk.ac.nesc.toe.sfk.radar.Scanner Condor #1 #2 #3 #4 #5
should_transfer_files = YES
when_to_transfer_output = ON_EXIT
Queue
```

---

### Figure 6: A script for submitting Globus jobs

This script submits Globus jdl's for execution

---

```
#!/bin/bash
source /opt/GT4/etc/switch-gt2.4-to-gt4.sh
for ((i=$1;i<=$2;i+=1)); do
    echo submitting job $i
    globusrun-ws -submit -F
    https://dc01.nesc.ed.ac.uk:8443/wsrp/services/ManagedJobFactoryService -S -f
    rsl/globus$i.rsl
    sleep 10
done
```

---

### Figure 7: A script for submitting Condor jobs

A script to submit Condor jobs for execution

---

```
#!/bin/bash
for ((i=$1;i<=$2;i+=1)); do
    echo $i
    condor_submit condor$i.jdl
    sleep 10
done
```

---