Comparing GRAM 2 and GRAM 4 in the Open Science Grid

Tino Vázquez, Eduardo Huedo, Rubén S. Montero, and Ignacio M. Llorente

Departamento de Arquitectura de Computadores y Automática
Facultad de Informática
Universidad Complutense de Madrid, Spain
{tinova, ehuedo}@fdi.ucm.es, {rubensm, llorente}@dacya.ucm.es

1 Introduction

The Open Science Grid[1] relies on the Globus Toolkit[2] to provide a grid middleware. Since the Globus Toolkit 4.0 was released, it split available resources in pre web services Globus and the latest web services implementations, that is, GT2 and GT4 respectively. The GridWay metascheduler offers full interoperability between both, so it presents itself like a good candidate to perform the task of benchmarking between both globus versions. This comparison shows the performance of the GRAM service under stress testing, under the point of view of the application (productivity).

2 GridWay Configuration

GridWay needed a few changes in its adaptor to to be correctly configured to interact to the OSG and thus perform this comparison. This changes were included in the 5.2.1 release of GridWay. There are integration guides for GridWay for the Open Science Grid[3].

One important thing to note is that for this tests GridWay needs to be compiled with pre.ws support. To have it's pre web service components available it is compulsory to use the ./configure flag "–enable-prews", since it is disabled by default.

The most significant changes needed in GridWay were:

- Modifications to the execution and transfer adaptors to be able to deal with non-standard globus ports. This is due to the fact that OSG Globus installations doesn't follow the standard 8443 port but rather the 9443 one.
- Unfortunately the information provided by mds2 in tOSG does not contain queue data and it is insufficient for gridway to be usable. Therefore, it is necessary to create static information for the nodes. For pre web service resources, the way to do this is to create a file for each of the nodes you

[1] http://www.opensciencegrid.org/
[2] http://www.globus.org
[3] http://www.gridway.org/documentation/stable/osghowto

want to configure with the needed information for the scheduler. This is an example of one of these files:

```
HOSTNAME="somehost.com" ARCH="i686" OS_NAME="Linux" OS_VERSION="2.4.21-32.ELsmp"
CPU_MODEL="Intel(R) Xeon(TM) CPU 2" CPU_MHZ=2665 CPU_FREE=189 CPU_SMP=2 NODECOUNT=1
SIZE_MEM_MB=2006 FREE_MEM_MB=964 SIZE_DISK_MB=73964 FREE_DISK_MB=62787
FORK_NAME="jobmanager-fork" LRMS_NAME="jobmanager-fork" LRMS_TYPE="fork"

QUEUE_NAME[0]="default" QUEUE_NODECOUNT[0]=1 QUEUE_FREENODECOUNT[0]=1
QUEUE_MAXTIME[0]=0 QUEUE_MAXCPUTIME[0]=0 QUEUE_MAXCOUNT[0]=1 QUEUE_MAXRUNNINGJOBS[0]=1
QUEUE_MAXJOBSINQUEUE[0]=0 QUEUE_STATUS[0]="0" QUEUE_DISPATCHTYPE[0]="Immediate"
```

. This file consists in two lines, one for the host general info and other for queue info. This information were gathered statically from the OSG Resources web page for each host. In this experiment we set the NODECOUNT to 10 to limit the number of jobs that GridWay is sending to each site. If you are adding more queues you have to add more lines, one for each queue. Variables will be named like

```
QUEUE_NAME[1], QUEUE_NAME[2]
```

and so on. Afterwards you will need to create another file with each line consisting in a pair of resource name and information file, the one previously described. This is an example of one of this files:

```
somehost.com etc/osg/somehost.com.attr
otherhost.com etc/osg/otherhost.com.attr
```

The path to the attribute files can be a full path or relative to `GW_LOCATION`. In gwd.conf you can use this lines as an example to add these resources:

```
IM_MAD = osg_mds2:gw_im_mad_static:-l etc/osg_prews.list:gridftp:osg_prews
EM_MAD = osg_prews:gw_em_mad_prews::rsl
TM_MAD = gridftp:gw_tm_mad_ftp:
```

– To configure WS OSG Resources you only need to have a file with a list of hosts you want to access, the information with this version of OSG is enough for gridway to schedule and sends jobs. For the purpose of this experiment, though, static information were used in the same fashion as above, but just setting the NODECOUNT to 10 to make the experiment fair for both interfaces. Therefore these lines were used in gwd.conf:

```
IM_MAD = osg_mds4:gw_im_mad_mds4:-l etc/osg_ws.list -p 9443:gridftp:osg_ws
EM_MAD = osg_ws:gw_em_mad_ws:-p 9443:rsl2
TM_MAD = gridftp:gw_tm_mad_ftp:
```

These peculiarities can be seen in Figure 1.

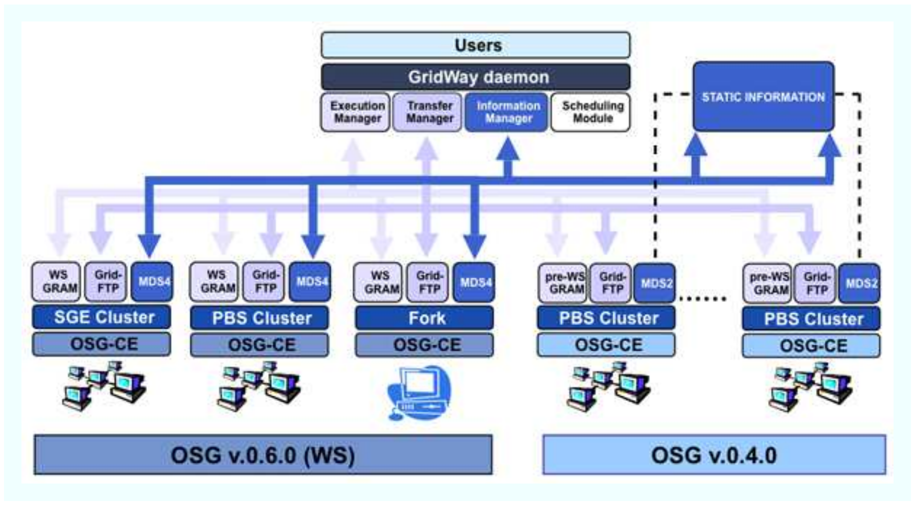

Fig. 1. Open Science Grid details.

3 Experiment Explained

For this experiment we used the Embarrassingly Distributed (ED) benchmark from the NAS Grid Benchmark (NGB) suite. This benchmark represent the class of Parameter Sweep Applications which represents a type of applications widely used in the grid.

The experiment is based on two different measures, one with a GRAM 2 (pre web services) based testbed, and the other with a GRAM 4 (web services) testbed. For each one, several measures were taken launching several hundred jobs for each one. GridWay was configured so it sends as much jobs as it can to each site.

As we have access to the OSG VO, we used the next two sites to conform our testbed configuring gridway as explained in the section above:

– osg-itb.ligo.caltech.edu
– nest.phys.uwm.edu

These sites have both GRAM 2 and GRAM 4 interfaces.

Once GridWay is configured with each testbed, two hundred jobs were launched. Each site was configured so only ten slots were seen by GridWay as available. This ensures a fair comparison between both experiments, because these two sites have a larger number of free slots, so the comparison doesn't depend on their workload (they are part of a test VO and, as far as these tests were concerned, never were so overloaded as not to have ten free slots).

To gather the results all we have to do is take a snapshot of the gwps output once the experiment is over (that is, once all the jobs have successfully completed), which gives us information about the start and end date of the job, it's successfulness, and it's transfer and execution time. This is all we need to perform the comparison.

In the next section we present a chart showing GRAM 2 against GRAM 4 productivity.

Also, a snapshot of the state of the jobs were taken each thirty seconds. The possible jobs states are:

– Pending - The job hasn't been scheduled or sent to a resource yet
– Prolog - The input files are being staged
– Wrapper Pending - The job has been sent and is waiting in the queue of the LRMS
– Wrapper Active - The job is running in the final computing element
– Epilog - The output files are being staged
– Done - The job has finalized

To capture this snapshot, a script was running each thirty seconds that, using gwps, counted the number of jobs in each of these states and saves the information onto a file with a time-stamp.

4 Results

In the productivity chart shown in Figure 2 graphic we have two different components, one for GRAM 2 (blue) and one for GRAM 4 (red). Each one measures how the number of jobs per hour varies in time. For that we plot for each job the TTBJ (time taken by the job, i.e., the time elapsed between its start and end time) divided by the number of jobs completed up to this one (included) against the TTBJ.

We can observe that productivity is slightly better in GRAM 4. The experiment for the Web Services GRAM finished earlier, and we can see that at the end of it productivity for GRAM 4 is about 15 jobs per hour more than that of GRAM 2.

Is worth noting at this point that, while productivity are around the same level, GRAM 4 offers much more in terms of scalability and also offers all the advantages of a Service Oriented Architecture, some of them being:

– Completely loosely coupled approach
– Authentication and authorization support at every level
– The search and connectivity to other services is dynamic

The other set of two charts were produced out of the data gathered in this experiment. In them, the number of jobs in each state are plotted against the time elapsed since the beginning of the experiment. We can observe how the number of active (running) jobs (the orange bit of the chart) maintains a constant figure. This is the result of the experiment, controlled so only 10 slots of each host are seen. In a real world scenario, this won't be as regular, it will depend on the workload of the sites, but this situation won't make a fair comparison between

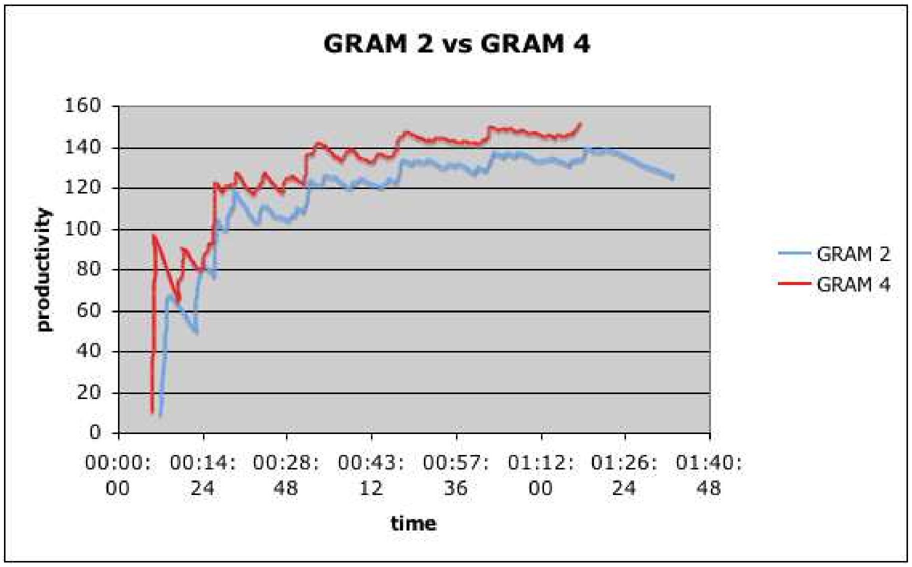

Fig. 2. GRAM 2 vs GRAM 4 - Productivity.

the two interfaces. There is not a significant difference between the two charts (Figure 3 and Figure 4), which we interpret as a marker for the fairness of the experiment

It was the intention of this report to also show the error rate of this experiment. No good chart can be plotted in this experiment out of this data because not a single job failed, and what is more, there was no need to migrate any job. This speaks not only for GridWay reliability but also for a good configuration and maintenance of the OSG resources we used.

If we want to look a bit deeper into the reason why the productivity in GRAM 4 is better than in GRAM 2 we have to calculate the overhead induced by each interface. We are going to calculate the *mean* times using all the jobs. This times and their meanings , i.e.:

- TOTAL time - Time elapsed between job submission and job completion
- SUSPENSION time - Time elapsed between when a job is submitted and when it actually starts running
- ACTIVE time - Time elapsed between when a job starts running and when it is completed as seen by GridWay
- REAL time - Actual time a job spends running. This information is gathered from an instrumentalization of GridWay's wrapper.
- OVERHEAD time - The overhead induced by the middleware. It is calculated adding the SUSPENSION plus ACTIVE times and then subtracting the REAL time.

If we take a look at Table 1 we can see that the overhead caused by GRAM 2 is 31 seconds, while GRAM 4 has an overhead of 24. Therefore we can conclude that this lesser overhead is the cause of the higher productivity.

	GRAM 2	GRAM 4
TOTAL	445	413
ACTIVE	410	389
SUSPEN	34	23
REAL	413	388
OVERHEAD	31	24

Table 1. Mean times for each experiment.

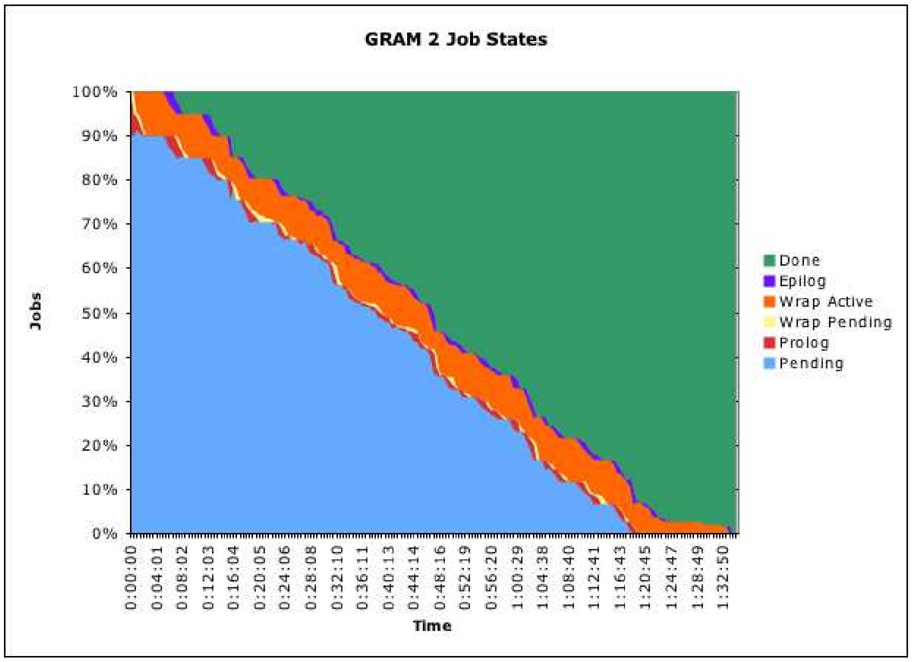

Fig. 3. GRAM 2 Job States.

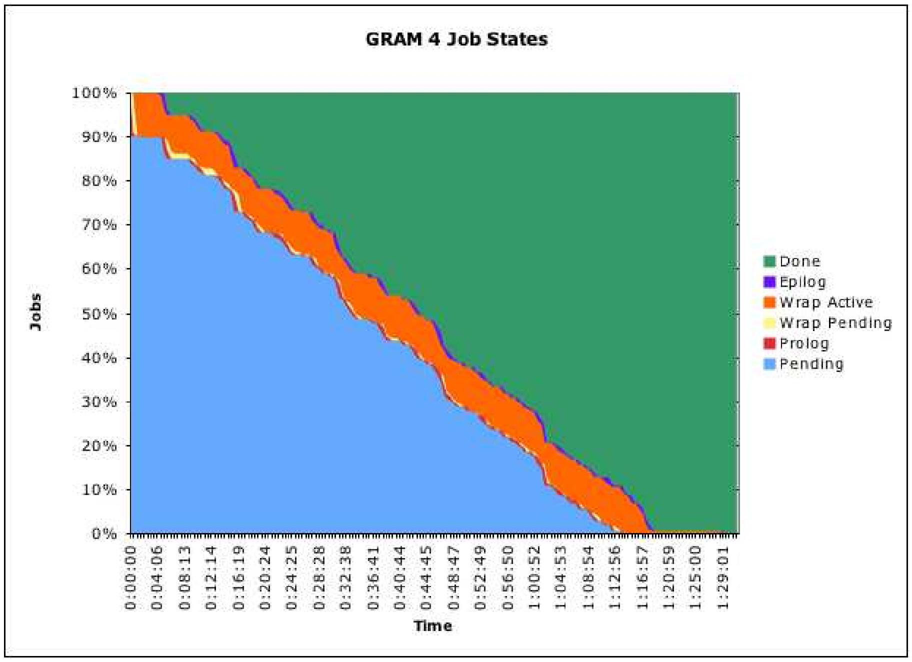

Fig. 4. GRAM 4 Job States.