

AliBaBa: Running BaBar jobs on the grid using gsub

AFS Access to local installations of BaBar

M.A.S Jones, Roger J. Barlow, Alessandra Forti
The University of Manchester

We have created a lightweight (server and client) intuitive command line interface through which users can submit jobs with complex software dependencies. We have added functionality that is specific to Babar jobs. gsub solves the input/output sandbox issues by the use of a global file system, currently AFS. Alibaba uses Gridsite to keep track of each job and display the status of sites in that grid.

gsub and Alibaba together provide a simple, easy-to-use grid submission and monitoring infrastructure that works and is currently being used by students and even professors for Particle Physics Analysis.

1 Introduction

Babar[1] is a B Factory particle physics experiment at the Stanford Linear Accelerator Center. Data collected here are processed at four different Tier A computing centres round the world: SLAC (USA), CCIN2P3 (France), RAL (UK), Karlsruhe (Germany).

Data is distributed to and research takes place at several institutes in the UK, all with designated Babar computing resources: the Rutherford Appleton Laboratory and the universities of Birmingham, Bristol, Brunel, Edinburgh, Liverpool, Manchester and Imperial College, Queen Mary and Royal Holloway, University of London. With this number of sites a grid is the obvious way to manage data analysis.

The ideal view of job submission to a grid in this context is one where the user prepares an analysis code and small control deck and places this in an 'input sandbox'. This is copied to the remote computer or file store. The code is executed reading from the input and

writing its results to the 'output sandbox'. The output sandbox is then copied back to the user.

Babar is a real particle physics experiment, involving several hundred physicist users, that has accumulated many millions of events, filling many Terabytes of storage. This simple model does not match the actual analysis of Babar data in several respects. This is partly due to the simplicity of the model and its inadequacy to describe real world computing use patterns. It is also partly due to the history of Babar, as its pattern of working evolved in a pre-Grid, central computing environment.

Firstly, a "job", as in a piece of work, will be split into hundreds (or even thousands) of "jobs", in the batch system sense of the term. On input a BaBar job needs to access the large event data files, which should be mounted locally to the platform the job is running on. Much of 'analysis' is filtering: an event is read, some (not many) calculations are done, some histograms may be updated and n-tuples filled, and the processing moves

on. It also needs to access many runtime files containing control parameters and processing instructions. This control has been implemented using tcl, and the files are known as the “*.tcl files” – .tcl files source more .tcl files, and the programs need other files (e.g. efficiency tables), dependencies quickly get out of hand. A general Babar user prepares their job in a ‘working directory’ specifically set up such that all these files are available directly or as pointers within the local file system, and the job expects to run in that directory.

One proposed solution for the job's input requirements is to roll up everything that might be needed in the 'inputsandbox' and send it: dump[2]. This will produce a massive file and you can never be sure you've sent all the files that the job might require. Another is to require all the 'Babar environment' files at the remote site, but that restricts the number of sites available to the user.

The job produces an output log - but this is usually of little interest, unless the job fails for some reason. The output that matters is in the histograms, which are written out to a binary file during and after the processing. To send these to the user by email is only sensible for small files and the Babar outputs can be quite large (if it includes ntuples for further analysis.) On the other hand one could expect the user retrieve the output themselves this requires the user or their agent knowing where the job actually ran. Another is for the job to push the output back itself, requiring write access (even a password) to the client's machine.

2 Babar and the Grid Overview

The UK BaBarGrid is data centric, it makes more sense to move execution to the datasets than the other way round. To this end the scientist uses skimData[3] to query a metadata catalogue storing event records and their location to find data. A web based interface to skimData also exists[4].

Using information about the data's whereabouts the user would normally be left to submit jobs and retrieve results at various sites manually. Alibaba allows users to submit their jobs to the compute nodes with access to nearby data and retrieve the output automatically as if the scientist was only dealing with local compute and data resources. Furthermore the scientist does not have to worry about software versions installed as their favourite version is transparently mounted on the remote resource.

3 Use of Existing Grid Infrastructure

Alibaba has two main aims. Firstly we aim to enable scientists to access resources with minimal impact to their normal mode of operation. Secondly we want to make the life of the system administrator as easy as possible. Alibaba uses existing core middleware installations and carries with any necessary baggage.

The Babar Virtual Organisation

Authority to run jobs on BaBar computational resources in the UK is determined either by the local system administrator or by membership to the BaBar Virtual Organisation. Membership to this VO is controlled by

people who pass the same criteria which govern access to the Babar Computing resources at Stanford. This is achieved by placing a file with details of the user's X.509 identity certificate in the user's home directory which is in the *slac.stanford.edu* Andrew's File System realm. A script runs regularly to pick up these files and check that the user is also in a specific AFS group (for which the user was required to sign conditions of use). The script transfers this information to an X.500 directory which can be accessed via the LDAP.

Globus Toolkit 2.x

Alibaba Job submission makes use of GRAM and GSI as provided by the Globus toolkit. Many of the sites involved have already got a Globus gatekeeper running and trusted root certificates and user lists.

EDG/GridPP Mapfile control

To configure the Babar Farms to authorise their use by these people and their jobs all that is required is for the Farm's system administrator to add a script to query this database via LDAP

and adjust the Globus grid-mapfile accordingly. One such script is *mk-gridmapfile* created for EDG[5] and LCG1[6].

4 Alibaba Middleware

The Alibaba solution is to use AFS in place of the sandboxes. *gsub*, a command with the look and feel of a batch-system submission command, effects this and hides most of the grid details. This saves the user the bother of having to collate all the necessary input beforehand and retrieve the data afterwards. Figure [1] shows the AFS solution.

A wrapper script is created automatically, staged and executed using *globus-job-submit*. When the script starts on the remote resource it prepares and maintains the jobs execution environment. It then forks a monitoring process, executes and becomes the job itself.

Access to the resource is based on VO management discussed above but only at

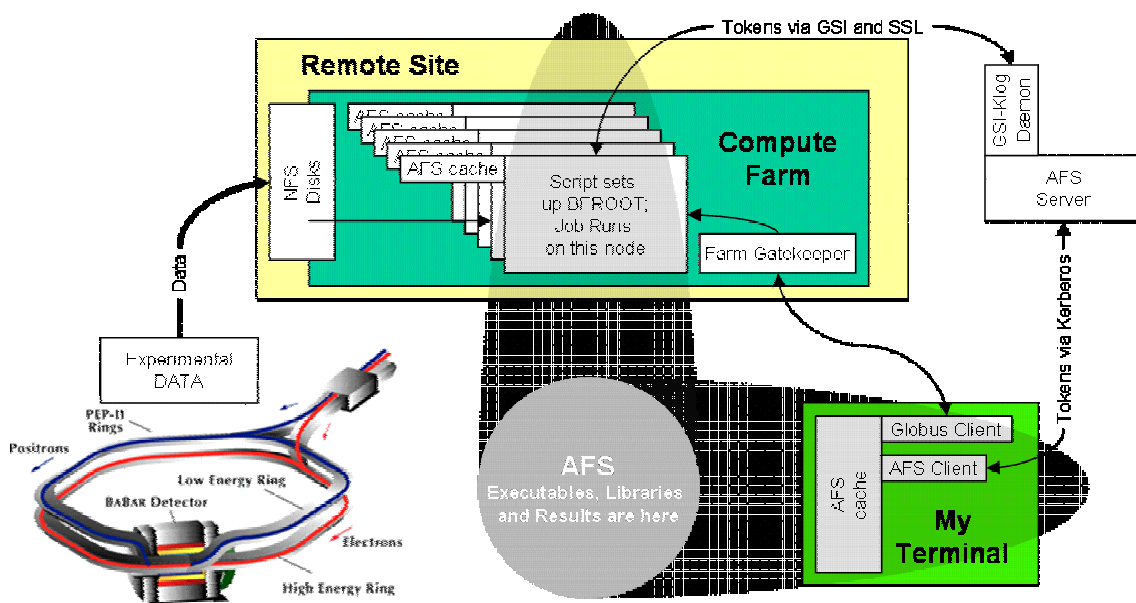


Figure 1 Schematic of an AFS based sandbox for grid submission

the Gatekeeper's end. The wrapper script executes `gsiklog`[7] to the home AFS cell. `gsiklog` is a self-contained binary that obtains an AFS token over SSL based on the authorisation of a grid certificate so an AFS password is not required. This executable is available to the job wrapper in a publicly accessible AFS directory and its integrity is checked by the evaluation of a cryptographic signature.

With an AFS token the job can then `cd` to the working directory (in the AFS). Input files can be read and output files can be written just as if running locally. Figure 1 shows the basic topology that `gsub` works to.

`gsub` is run (just like submitting to a familiar batch system) as follows:
`gsub [options] command arguments`

`gsub` does the following, it:

1. checks the executables and grid credentials etc.;
2. gets the current list of gatekeepers;
3. creates a script (to wrap the executable on remote batch node);
4. uses Globus to stage and submit the script to a queue on a local/remote machine;
5. uses `curl` over GSI-authenticated SSL to tell a website the status of the job;

The script created by `gsub` is staged through the Globus interface to the remote batch system and does the following:

1. sets up a normal environment[8]. Globus by default does not provide a sensible execution environment, it must be built at run time;
2. notifies a website for monitoring;
3. gets (pag separated) AFS credentials using `gsi klog`. Each job runs with access to its own AFS credential so that it may be removed when the job

- finishes;
4. creates `BFROOT` – the Babar directory system and replicates the Babar environment;
5. changes to directory submitted from in AFS;
6. starts a shepherd process (this will look after job's grid credentials and talk to the website and `MyProxy`[9] server if required);
7. runs user's executable (script or binary);
8. unlogs (removes credentials no longer required).

Monitoring the Job and the Grid

`gsub` submitted jobs upload progress information to a Gridsite[10] (a grid enabled Apache webserver). Information handled this way is uploaded securely (XML files over `https`). Authorisation to upload information is based on the standard mutual authentication of an `https` server but using the GSI proxy credential available to the running job rather than a standard X.509 certificate. Only credentials issued by the owner of the job can alter job status files on the webserver after the initial job submission.

Information about the Job is available in parts to the general public via HTML (see figure [2]). The job's owner is able to access more information by visiting the same website and authenticating using their X.509 certificate (figure [3]). The information available to the authenticated user is:

- the times at which the job's status changed,
- the IP address of the batch node,
- the x-gram url of the gatekeeper,
- the UID of the submitter,
- the UID as mapped on the batch node,

Contact URL	Stages					Exit Status	Action
	Submitted	Confirmed	Started	Running	Finished		
https://bfb.hep.man.ac.uk:50103/14701/1080753881/	Wed Mar 31 17:24:29 2004	+0:00:15	+0:00:53	+0:00:55	+0:01:27	0	
https://bfb.hep.man.ac.uk:50100/14637/1080753863/	Wed Mar 31 17:24:14 2004	+0:00:14	+0:00:31	+0:00:35	+0:01:05	0	
https://bfb.hep.man.ac.uk:50095/14453/1080753840/	Wed Mar 31 17:23:51 2004	+0:00:22	+0:00:52	+0:00:55	+0:01:25	0	
https://bfb.hep.man.ac.uk:50074/14325/1080753826/	Wed Mar 31 17:23:37 2004	+0:00:12	+0:01:05	+0:01:09	+0:01:39	0	
https://bfb.hep.man.ac.uk:50073/13908/1080753763/	Wed Mar 31 17:22:31 2004	+0:00:31	+0:01:11	+0:01:13	+0:01:43	0	
https://bfb.hep.man.ac.uk:50100/14117/1080753794/	Wed Mar 31 17:23:04 2004	+0:00:32					
https://bfb.hep.man.ac.uk:50096/13752/1080753741/	Wed Mar 31 17:22:13 2004	+0:00:17	+0:00:37	+0:00:39	+0:01:10	0	
https://bfb.hep.man.ac.uk:50074/13573/1080753715/	Wed Mar 31 17:21:46 2004	+0:00:25	+0:01:04	+0:01:06	+0:01:37	0	
https://bfb.hep.man.ac.uk:50072/13074/1080753655/	Wed Mar 31 17:20:43 2004	+0:00:23	+0:00:55	+0:00:58	+0:01:28	0	
https://bfb.hep.man.ac.uk:50093/13262/1080753679/	Wed Mar 31 17:21:08 2004	+0:00:36					
https://bfb.hep.man.ac.uk:50100/12657/1080753618/	Wed Mar 31 17:20:08 2004	+0:00:13	+0:00:48	+0:00:52	+0:01:28	0	

Figure 2 Alibaba's display of public job information

- the exit status of the job,
 - the Globus ID of the job,
 - and the time remaining of the proxy.
- This information allows the user to orchestrate their job submissions and pick up errors as they occur. It also allows the monitoring of proxies which may need updating (either automatically via MyProxy) or manually through Globus.

One further advantage of uploading the jobs' statuses to a single server is that this information can be used to present an overview of the grid. Figure [3] shows a status map of the grid enabled Babar resources in the UK. The average time to execution is available and the numbers of jobs in their various stages can be seen. This information can be used by the administrators for maintenance purposes and by the

Contact URL	Stages					Exit Status	Action
	Submitted	Confirmed	Started	Running	Finished		
https://bfb.hep.man.ac.uk:50103/14701/1080753881/	Wed Mar 31 17:24:29 2004	+0:00:15	+0:00:53	+0:00:55	+0:01:27	0	
https://bfb.hep.man.ac.uk:50100/14637/1080753863/	Wed Mar 31 17:24:14 2004	+0:00:14	+0:00:31	+0:00:35	+0:01:05	0	
https://bfb.hep.man.ac.uk:50095/14453/1080753840/	Wed Mar 31 17:23:51 2004	+0:00:22	+0:00:52	+0:00:55	+0:01:25	0	
https://bfb.hep.man.ac.uk:50074/14325/1080753826/	Wed Mar 31 17:23:37 2004	+0:00:12	+0:01:05	+0:01:09	+0:01:39	0	

Figure 3 Alibaba Information for the authorised

scientists to help them decide where to run jobs.

5 Alibaba via Ganga

gsub has been modified to split the job preparation and submission components to allow Ganga[11] to interface with it. A Graphical User Interface has been constructed within this framework and is shown in figure [4].

6 Conclusions

The Alibaba solution has the following advantages -

Minimal requirements at the remote host site. The gatekeeper has to have Globus and some batch system. The worker nodes do not require any Babar software (though the operating system has to be

compatible with the binary).

They do not require to have gsiklog, as it is supplied.

They do have to have AFS (client) , and to have IP communication with the outside world. Given the widespread use of AFS, this is not a harsh requirement. Whether worker nodes at future computer centres will or should have IP access is being hotly debated at the present time: the centres are reluctant but the users are demanding it.

Minimal user changes; the user executes commands in their 'workdir' directory just as with the original central computing model system. Whatever is available in that directory is available to the remotely running job. The output histogram files appear in that directory

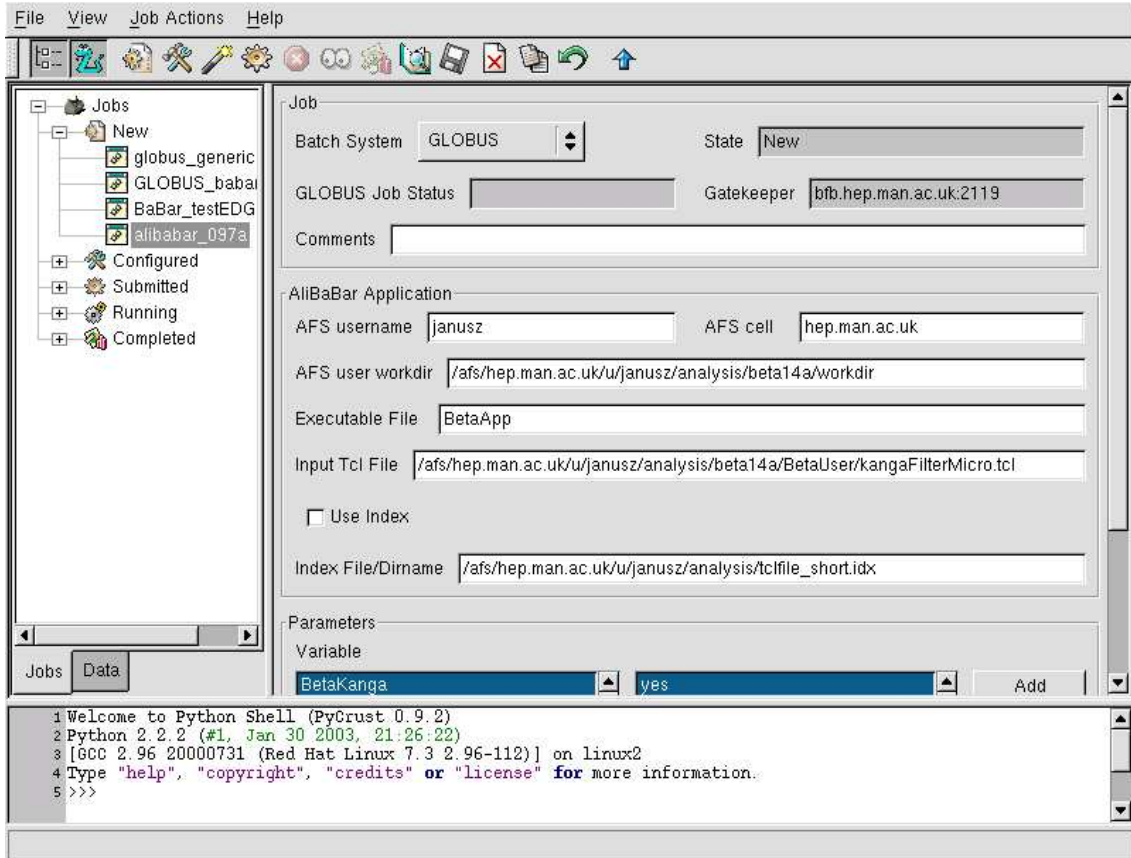


Figure 4 Ganga Interface to Alibaba

just as they do if a job is running on the local system.

Alibaba provides a framework for easy job submission, grid monitoring and the tracking of jobs; three key requirements when moving from the desktop to the great blue yonder.

7 Acknowledgements

Thanks to Janusz Martyniak Imperial College for work on the Ganga implementation of gsub. Thanks also to Andy McNab for installing the Gridsite that hosts the Alibaba website.

8 References:

1. The BaBar homepage:
<http://www.slac.stanford.edu/BFROOT>
2. Boutigny et. Al. on behalf of the Babar Computing Group, (2003) Use of the

- European Data Grid software in the framework of the BaBar distributed computing model. Conference Proceedings CHEP03 La Jolla
3. BBRORA and skimData
<http://www.slac.stanford.edu/BFROOT/www/Computing/Offline/DataDist/KDDIBT.html>
 4. skimData Web Interface
<http://bfhome.hep.man.ac.uk/skimData.pl>
 5. EDG <http://www.eu-datagrid.org>
 6. LCG <http://lcg.web.cern.ch/LCG>
 7. gsi enabled klog
<ftp://achilles.ctd.anl.gov/pub/DEE/gssklog-0.11.tar>
 8. HEPiX Shell Scripts Files
<http://wwwhepix.web.cern.ch/wwwhepix/wg/scripts/www/shells/files.html>
 9. MyProxy <http://grid.ncsa.uiuc.edu/myproxy/>
 10. Gridsite <http://www.gridpp.ac.uk/gridsite/>
 11. Gaudi/Athena and Grid Alliance
<http://ganga.web.cern.ch/ganga/>