

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.

Introduction

The ideal view of job submission to a grid 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.

This simple model poses some problems for analysis of BaBar data. On input the BaBar job needs many runtime files “*.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 'input sandbox' and send it. 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.

As for retrieving output this could be emailed to the user – but this is only sensible for small files and the BaBar outputs can be huge (it can be 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.

Submitting Jobs With gsub

Our solution is to use AFS. The job is executed at a remote site using globus-job-submit. Access to the resource is based on VO management discussed above but only at the Gatekeeper's end. A simple wrapper round the job executes gziklog to the home AFS cell. gziklog is a self-contained binary that performs a klog based on the authorisation of a grid certificate so an AFS password is not required. The job can then cd to the working directory in 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
```

1. checks 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 ssl to tell a website the status of the job (alibaba)

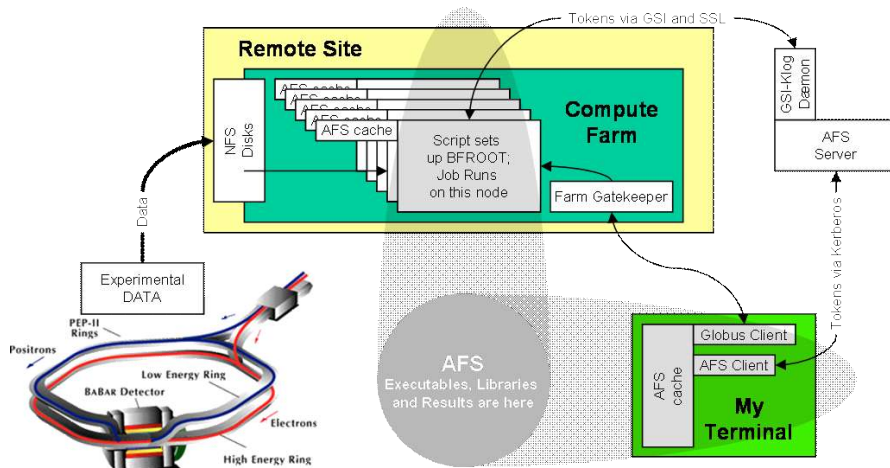


Figure 1. Schematic of an AFS based sandbox for grid submission of analyses

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^[1]
2. notifies alibaba (the gridsite based monitor)
3. gets (pag separated) AFS credentials using gsi klog
4. creates BFROOT – 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 alibaba)
7. runs user's executable (script or binary)
8. unlogs (removes credentials no longer required)

Monitoring And The AliBaBa Web Page

Using Gridsite^[2] (an apache webserver with modifications to amongst other things accept GSI proxy credentials) gsub submitted jobs can upload progress information. Information handled this way is uploaded securely (XML files over https) and is available in parts to the general public via HTML. The job's owner is able to access more information by visiting the same website and authenticating using their X509 certificate. Figure 2 shows the website interface to data uploaded this way.

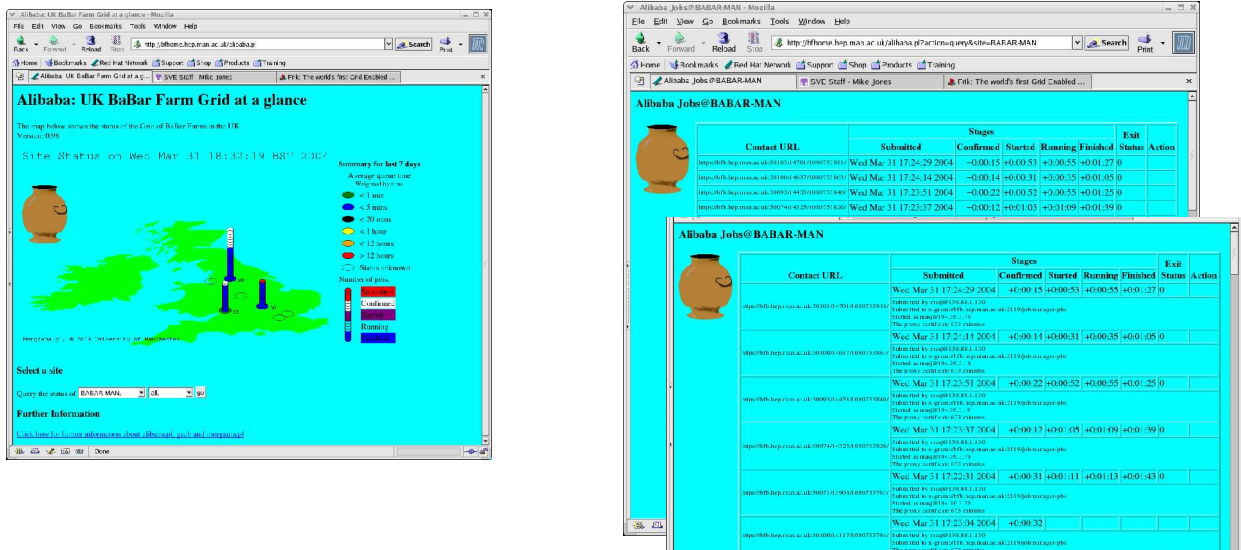


Figure 2 AliBaBa web user interface

[1] HEPiX Shell Scripts – Files <http://www.hepnet.org.uk/~wwwhepnet/wg/scripts/www/shells/files.html>

[2] Gridsite <http://www.gridpp.ac.uk/gridsite/>