

THE BABAR GRID DEMONSTRATOR PROJECT

T. Adye, R Barlow, A Forti, A McNab D Smith on behalf of BaBar and GridPP

Key words to describe the work: Authorisation, Authentication, MetaData, Remote Job Handling

Key Objectives: Building a system to enable members of the BaBar experiment to run analysis jobs at all participating sites

Motivation for the work (problems addressed):

- 1) Need to locate data (spread across several sites, with multiple copies) according to user specifications.
- 2) Need for the user to be able to run jobs at remote sites, without having every user registering at every site.
- 3) Need for the user to be able to run jobs at remote sites with the minimal requirements for the environment available.
- 4) Need for the user to have output returned in a convenient way without endangering security.

The BaBar experiment[1] is located at the Stanford Linear Accelerator Center[2], where it has been collecting data on the decay of B mesons since 1999. Many millions of decays have been recorded, each providing tens of kilobytes of data. The collaboration comprises 560 physicists from 9 countries. These physicists analyse the data using purpose-built programs which generally require a considerable processing time.

As the data volume increases the collaboration is moving away from a model in which computing is done at a single centre, where all data files and all programs are available, to a distributed model with many computing centres, of various sizes, with data spread across them.

Grid technology is clearly the way to make this work. In an ideal future system one can envisage a user specifying the type of data they want to analyse and how they want to analyse it, and a job submission system that locates the relevant datafiles (choosing between alternatives if there are multiple copies), runs the analysis job on those files using available local CPUs, retrieves the various outputs, combines them seamlessly, and returns them to the user.

The purpose of this demonstrator is to achieve as much of this as is possible today, as a first and hopefully useful step towards the future system. The demonstrator runs through a web browser, presumably running on the user's desktop or laptop. We assume that the user does not have any further software (such as globus) running on this platform, as this would be rather restrictive.

Data location is done through a metadata system. We do not yet have a single replica catalog that knows about all copies of all files. However each site knows what files it has, using the skimData system[3], and can respond remotely to queries. The user makes their specification through a convenient

web form. They then provide the sites they want to run at, in an ordered list. A web engine locates the matching data available at the first site using the skimData remote query facility. It then enquires of the second site for matching data which was not at the first site, and this is repeated through the site list. The jobs are then prepared for submission. The total process (or hyperjob) is given a unique jobid of the form `bbargridfilesnn`. The hyperjob involves several sites. At each site the process (or superjob) is split into several separate single jobs. This is done by the user specifying how many events can be handled in a single job, and the data is divided into separate jobs accordingly. The number of jobs submitted to a site should be matched to the number of separate CPUs available. This is another point where at present the user's judgement is required and in future a resource broker will make such decisions. For each superjob, there is a Job0 in which the required binary executable is copied to the remote site, as are various other essential files. Then the individual jobs follow.

Job submission is done by a cgi perl script running in a web server (at present the web server for GridPP.) This submits jobs to the remote sites using `globus-job-submit`. The user creates a grid-proxy and then uploads it into the server. This provides a single entry authorisation point, as the browser then uses this certificate to authenticate the globus job submission.

Authorisation is arranged by using the certificate to identify the user and comparing that with a list. The jobs running at a remote site are submitted through a gatekeeper, typically to a batchsystem such as `pbs` running on a farm. The gatekeeper will attempt to match the name on the grid certificate used to submit it to a name in its `.globus/emap` file. If the user has made appropriate arrangements then an entry

exists, but in general one cannot expect each of 560 users to make arrangements at each of 50+ sites, and the site managers are also reluctant once this is explained to them. (Some of the sites are for the exclusive use of BaBar but some are general national facilities for HEP.) A system of dynamic accounts is therefore used[4]. Any BaBar grid user has by definition a Grid certificate from an accepted authority, and an account on the central SLAC system with BaBar authorisation in the afs acl list. They can register for BaBarGrid use merely by copying their grid name into a file in their home area. A cron job then picks this up and sends it to the central BaBar VO machine (currently at Manchester) from where all participating sites pick up the list of authorized Babar users and put it into their map files, using another cron job, giving the generic userid .babar.

This system has proved easy to operate and reliable. The map files are handled in sequence such that if a user has a personal account at a site this is encountered first and therefore used in preference to the generic account. The cron job that updates the map files is under control of the local system manager who retains the power to modify it (perhaps by removing certain users?) and thus retains control. It would be straightforward to ensure that these generic accounts had low levels of privilege, and thus local users can be given priority over ones from outside.

For each job one requires:

- ⌘ The binary
- ⌘ The data files
- ⌘ A .tcl file specifying all the data files for this job.
- ⌘ A small .tcl file that pulls in the others
- ⌘ A large .tcl file containing standard proceural stuff
- ⌘ Various other data files
- ⌘ The calibration (conditions) database
- ⌘ The setting of appropriate environment variables
- ⌘ The presence of some dynamic (shared) libraries

This is generally known as the 'input sandbox' and shows the increasing demands once one goes beyond running a simple 'Hello world' job. For Babar this is a particular problem as the software was developed before the Grid: it is assumed that the job is run in a 'release directory' in which all these files are made available.

Our solution is to provide these files and ship them Alternatives are

1. Only to run at sites where the desired release directory is available. We wanted to have something less restrictive
2. To run from within an afs directory and to klog and cd to that directory as the very first step of each job. This requires the job to contain the afs password which is bad security, though the gsiklog system will avoid this.
3. The Condor solution of directing all I/O through the submitting machine. Unfortunately this would not work for the access of the actual event data. Future enhancements to Condor that will allow both types of access will make this more attractive.

The job runs, and the standard output is available through an url that point to the gass-cache. The submission system catches this url and provides a link to it for the user.

Actually in most cases this output is not of any interest (unless things go wrong.) The useful data is stored as histograms and/or ntuple. It will be produced on files at the remote site, on the gatekeeper. For convenience, as each job finishes it moves its output file to a central directory /tmp/<jobid> and then for convenience tars together all the files there.

The user can then request from a web form that the outputs are collected. This is done on a 'local' machine which has to have spare disk space and be running grid-ftp. A job is submitted to this machine which copies the files from each site (superjob), untars each and then copies them all to one directory (for the hyperjob) which is again tarred.

A link is provided and a specific MIME type given. When the link is clicked on the directory is downloaded to the desktop machine where the application specific to this type has been arranged to unpack the directory and run a standard analysis job on it to draw the desired histograms.

The demonstrator is operational and links the BaBar UK sites, and is being extended to sites in France and Germany. It is planned to extend its functionality by permitting the user to specify the binary file to run the analysis and the final job to process the histograms; this will take it from being a demonstrator to being a useful tool

References

[1] BaBar

[2] PEP-II

[3] skimData

[4] dynamic accounts