

Distributing file-based data to remote sites within the *BABAR* Collaboration

*Tim Adye*¹, *Alvise Dorigo*², *Alessandra Forti*³, *Emanuele Leonardi*⁴ on behalf of the *BABAR* Collaboration's Computing group

¹Rutherford Appleton Laboratory (UK), ²INFN Padova (Italy), ³Manchester University (UK), ⁴INFN Roma (Italy)

Introduction

BABAR [1] uses two formats for its data: Objectivity database and ROOT [2] files. This poster concerns the distribution of the latter — for Objectivity data see [3].

The *BABAR* analysis data is stored in ROOT files — one per physics run and analysis selection channel — maintained in a large directory tree. Currently *BABAR* has more than 4.5 TBytes in 200,000 ROOT files. This data is (mostly) produced at SLAC, but is required for analysis at universities and research centres throughout the US and Europe.

Two basic problems confront us when we seek to import bulk data from SLAC to an institute's local storage via the network. We must determine which files must be imported (depending on the local site requirements and which files have already been imported), and we must make the optimum use of the network when transferring the data. Basic ftp-like tools (`ftp`, `scp`, etc) do not attempt to solve the first problem. More sophisticated tools like `rsync` [4], the widely-used mirror/synchronisation program, compare local and remote file systems, checking for changes (based on file date, size and, if desired, an elaborate checksum) in order to only copy new or modified files. However `rsync` allows for only limited file selection. Also when, as in *BABAR*, an extremely large directory structure must be scanned, `rsync` can take several hours just to determine which files need to be copied. Although `rsync` (and `scp`) provides on-the-fly compression, it does not allow us to optimise the network transfer by using multiple streams, adjusting the TCP window size, or separating encrypted authentication from unencrypted data channels.

Data Transfer Schema and File Management Tools

In order to retrieve the information about which files are missing at the local site we exploit **BBRORA**, an ORACLE-based database which was created at SLAC to keep track of data production. It is also used by physicists to select data for analysis, based on criteria like their physics content, and calibration or software release used to process the data. Information stored in **BBRORA** was easily adapted to use by our data-transfer tool and to keep track of the life-cycle of the locally imported files.

Mirror: The first step is to copy the relevant tables of **BBRORA** to a MySQL [5] database located at the regional centre: each database entry corresponds to a single data file produced at SLAC. Each entry already included the file path and size, checksum, number of events, software release used, etc. To these we added a special field named `status` to store the status of the file with respect to the transfer, backup, and deletion process. The local **BBRORA** is updated daily with a dedicated mirror tool: this task takes only a few minutes and initially sets the `status` field for each new entry to a special value, TO-BE-CHECKED, which means that the file is a candidate for import.

Database Management: When the mirroring task is over, a second tool queries the database for all entries marked as TO-BE-CHECKED, changing their `status` to TO-BE-IMPORTED if the file is needed for local physics analysis, or to NOT-TO-BE-IMPORTED if the file is not needed. An import priority can also be added to the `status` field. If the file is already on disk (eg. it

was copied manually), it is marked ON-DISK. These selections can also be changed under user control if the site requirements change.

File Import: When all the checks are done, the import step starts. The import tool obtains from **BBRORA** a list of files to be imported and copies them using a network-based multi-stream transfer procedure. The first step is quickly accomplished with a simple SQL query which looks for all entries marked as TO-BE-IMPORTED. The resulting list of file is then passed to the import tool which connects to SLAC using a secure channel (**ssh**) and transfers them (for speed, the data channels are unencrypted). The total time from the beginning of the mirroring step to the beginning of the file-transfer step is always less than 20 minutes, with a small dependency on the actual size of the list of files to be imported.

The import tool is a sophisticated wrapper to which several low level ftp-like protocols, both with or without multi-streaming capabilities and TCP window size control (**scp**, **bbftp** [6], **sftp**, and soon **bbcp** [7]), can plug-in. As an efficient use of the network link is mandatory in order to keep pace with data production, the wrapper has to carefully handle the total number of parallel threads used for file transfers.

When a file is successfully imported the corresponding **status** field is set to ON-DISK — it is just left unchanged if the transfer fails.

Using the database and the advanced file transfer tools allow us to import data at a rate limited only by the WAN bandwidth constraints (either the physical limit, or less if fewer streams are specified to limit the load on the network). For example, a sustained rate of about 30 Mbits/sec from SLAC to INFN Rome has been achieved.

Backup: An additional tool is used to handle file back-up to a tape storage system: this tool queries **BBRORA** for entries marked as ON-DISK and copies the corresponding files to tape, marking them as ON-DISK-AND-ON-TAPE. The relatively small ROOT files are grouped into larger files using **tar** before copying to tape. Since different sites have different tape systems (with different interfaces), local configuration procedures handle low-level details needed to map filename to tape position and local staging system control.

File Deletion and File System–Database Consistency: Two additional tools simplify data management at each site. One allows for a detailed selection of files to be removed from disk, marking them as ON-TAPE-ONLY. The other is used to make consistency checks between the content of the **BBRORA** archive and the actual status of the file systems on disk.

Users: The information contained in the local copy of **BBRORA** can also be used to quickly collect a whole set of useful statistics such as total disk or tape usage, number of files or events from a particular year, processing version, physics selection stream, and so on. Finally, given its homogeneity with the central **BBRORA** at SLAC, it allows local users to select the files for their analysis with the very same tools they use at SLAC.

References

- [1] *BABAR* Collaboration, B. Aubert et al., “The *BABAR* Detector”, SLAC–PUB–8569, to appear in Nucl. Instr. and Methods.
- [2] R. Brun and F. Rademakers, ROOT home page, CERN, <http://root.cern.ch/>.
- [3] D. Boutigny, “The *BABAR* Experiment Distributed Computing Model”, Oral Session at this conference — Abstract 4–021.
- [4] A. Tridgell, Chapters 3–5, “Efficient Algorithms for Sorting and Synchronization”, PhD Thesis, The Australian National University, April 2000; and <http://rsync.samba.org/>.
- [5] MySQL, a powerful open-source relational database system, <http://www.mysql.com/>.
- [6] G. Farrache, BBFTP home page, IN2P3, <http://ccweb.in2p3.fr/bbftp/>.
- [7] A. Hanushevsky, “Exploiting Peer-to-Peer Computing for Secure High Performance Data Copying” (**bbcp**), Oral Session at this conference.