PRODUCTION DATA EXPORT AND ARCHIVING SYSTEM FOR NEW DATA FORMAT OF THE BABAR EXPERIMENT

Artem Trunov, Tofigh Azemoon, Adil Hasan, Wilko Kröger
On Behalf of the BaBar Computing Group
SLAC Computing Services, Stanford, CA 94025, USA

Abstract

BaBar has recently moved away from using Objectivity/DB for its event store toward a ROOT-based one. Data in the new format is produced at over two dozen institutions worldwide including SLAC. Among new challenges is the organization of data export from remote institutions, its archival at SLAC and bringing it to the users for analysis or import to their own institutions. The new system is designed to be scalable, easily configurable on a client and server side and adaptive to server load. It is integrated to work with BaBar's mass storage system (HPSS) and with xrootd service [1]. Design, implementation, experience with the new system, as well as future developments are discussed in this article.

BABAR DATA PRODUCTION

The BaBar Experiment based at Stanford Linear Accelerator Center has been taking data since May 1999. Initial design of the event store utilized Objectivity/DB [2] as the database technology [3]. Later, a ROOT-based event store, called Kanga, was developed and used in parallel in analysis only. Since the beginning of Run 4 (Nov 2003), a new ROOT-based event store, was developed and used as the primary event store of the experiment [4]. Almost all the data stored in Objectivity/DB has been converted to the new format.

The volume of data accumulated in Objectivity/DB federations as of today is 931 TB. Over 161 TB are produced in the new Kanga format, including converted Objectivity data, initial reconstruction of Run 4 data, and skimming of Run 1-4 data.

Over 260 TB of disk space is used for production and analysis at SLAC.

Distributed Production

Since the beginning of the experiment, data has been produced not only at SLAC, but at other institutions. Initially, the Monte Carlo production was done at SLAC and LLNL. Later, IN2P3 (France) and several other institutions started to contribute to the Simulation Production. Today, with over 1 PetaByte of data, the production runs at over two dozen institutions in the US, Canada and Europe. Nowdays, off-site data production is not limited to simulation. INFN (Padova) does all the initial event reconstruction. IN2P3, INFN and GridKa (Karlsruhe, Germany) contribute to skimming.

Past Experience with Data Management

Distributed production efforts in BaBar brought some challenges in data management. Initially, data from IN2P3 was shipped to SLAC on tapes via commercial postal services. Shipment delays and manual handling of such imported data at SLAC, as well as improvements in network connections, led to switching to network only import/export. To improve transfer speed and network utilization bbcp [5], high performance multistream copy utility, was developed at SLAC.

Another complication of the import/export procedure was the specifics of Objectivity/DB OODBMS and the event store design based on it. Database files had to be registered in the Objectivity federation catalog. During this process, referred to as attaching a database to a federation, the database file was scanned by Objectivity/DB administration utility and any data corruption halted the process, requiring intervention of a database administrator. Also, new collections had to be loaded to the collection tree (application level catalog). Both operations required locking of the federation's critical metadata, which interfered with the users' analysis activities. Sometimes a stubborn lock from another application prevented the import from completion until the lock was removed. Scalability problem of Objectivity federation, which manifested itself in growing metadata access time as the federation got larger, also affected delivery of data to physicists.

To ease the burden of lock conflicts in analysis federations, import of off-site data was done in dedicated import federations, where databases were initially attached. Subsequently, databases were attached to the analysis federations without a new scan, reducing locking time. Collections were loaded directly to the analysis federations.

Nevertheless such a procedure was not used for SLAC production, where delivery time was more critical, and it was easier to solve all problems within the same site.

As mentioned above, data corruption was one of the major issues for data import since Objectivity/DB tools were limited to checking for corruption at a database level. The ollections could only be checked with standard physics analysis applications, which was unacceptable because of time and resources such operations required.

After importing into Objectivity/DB federation the data had to be archived in SLAC mass storage system (HPSS by IBM.) This system, while offering excellent reliability and scalability to over 1 PB of data, has its own disadvantages. HPSS uses a proprietary code, thus tape mount order and file read and write order could not be controlled. Additional
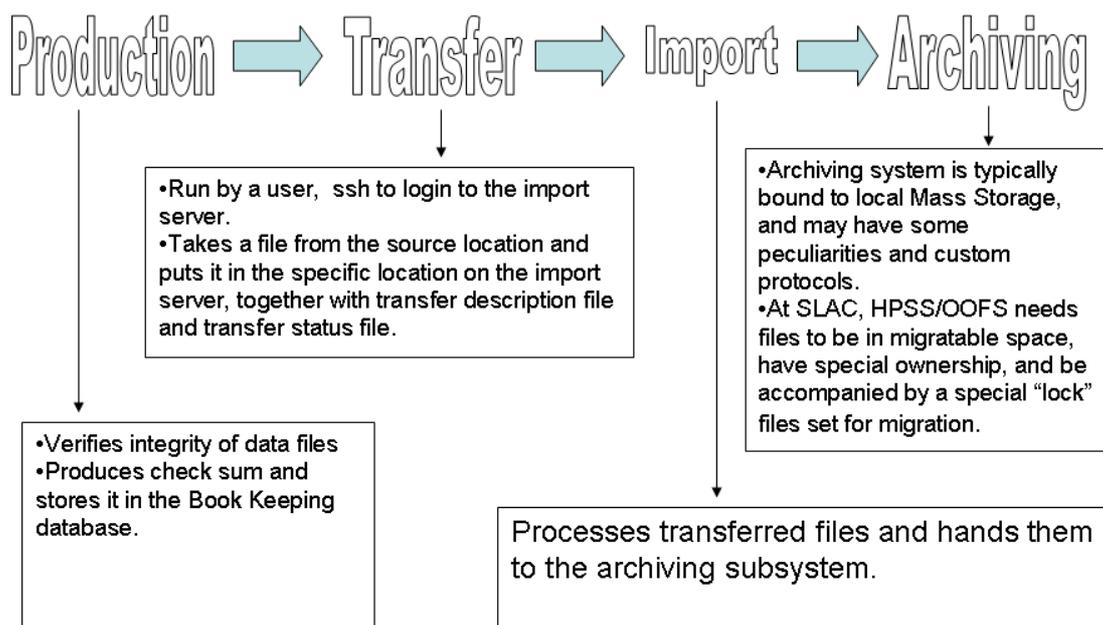
Figure 1: Sharing concerns and responsibilities between subsystems.

disk space had to be acquired to reduce the load on HPSS and the staging time.

To summarize, import/export problems included:

- Absence of efficient higher level tools to check data consistency
- Very little automation of import error handling
- Different import procedures for data produced at SLAC and at remote sites
- Inability to control HPSS to a desirable degree.

Motivation for New Development

Some of the problems metioned above have been addressed in the design of the new ROOT-based event store. Since there are no more federations, data files have only to be put in the mass storage and in the analysis area where users can access them. There are no corruption issues, because data consistency is checked with a special tool before leaving the production site. To take advantage of the new event store design and resolve remaining issues, new transfer/import tools were needed. Among the requirements for the new transfer tool were:

- Full automation of data transfer and archiving
- Unification of all export and archiving procedures
- Reduction of human involvement in error handling
- HPSS-friendly system
- Low resource utilization, with focus on disk bandwidth
- Protocol level backward compatibility with Objy transfer/import tools
- Streamlined procedure for further data processing
- Assisting with exporting data to remote institutions

DEVELOPMENT OF A NEW TOOL

During the development of the new tool, focus was not on designing yet another full-blown Storage Resource Manager (SRM) or metadata catalog, but on making it as simple as possible, while implementing the basic ideas discussed below.

Sharing Responsibilities and Concerns between Subsystems

The first idea is illustrated on Fig. 1. The goal is to modularize transfer/import and define responsibilities of each subsystem. In our schema, the **Production** subsystem is responsible for checking data for quality and consistency. If data corruption is detected at a later stage, this subsystem will have to deal with the issue.

Transfer subsystem is responsible for delivering of data from production site to the import servers. It verifies checksum after transfer and keeps the necessary metadata about transferred files.

Archiving subsystem is responsible for saving files in the Mass Storage System. It is usually very site-specific and has its own protocol and policies, making it difficult to use out-of-box tools.

The role of **Import** subsystem is very simple in this schema. It processes files after the transfer, preparing them for archiving and hands them over to the **Archiving** subsystem. Preparation may include changing file ownership, placing files in migratable space and complying with other MSS protocols.

It is worth emphasizing, that transfer is detached from import, making the whole procedure less susceptible to any failure that might occur during WAN transfers or even at the remote site.
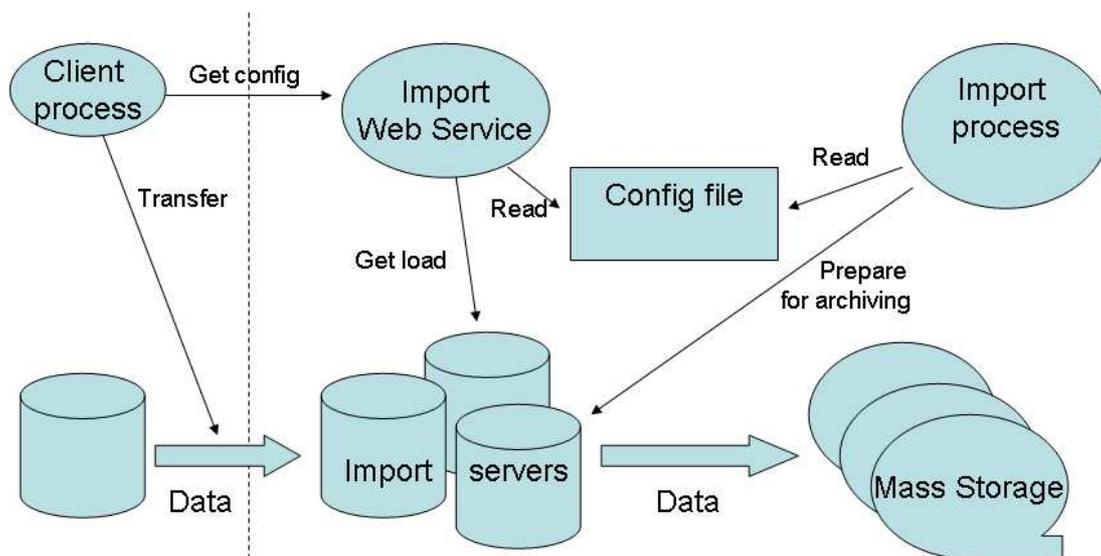
Figure 2: Central management.

Push-Pull model of data transfer

As SLAC is the only site that archives all the produced data in the Mass Storage, and provides all the data to end users, it would be natural to have a SLAC-centric data distribution model. Data management at SLAC is provided by the Computing Department. All services are expected to be always available and problems are expected to be solved in a timely manner. Therefore we have chosen a transfer model, where remote producers "push" data into SLAC. In this model, local problems at multiple remote institutions are no concern for SLAC administration, and only local administrators are dealing with them. However, a problem that occurs at SLAC, would most likely affect all data exporters and would be fixed by an administrator on duty.

On the other hand, the data that needs to be exported to a remote site for analysis by local physicists is "pulled" from SLAC. This is again done for the same reasons — sites initiate transfer when they are ready, and may choose to implement a protocol at their end that suits them.

Central Management

The idea of central management is illustrated in Fig. 2.

Here import is a web service that provides the clients with transfer parameters, like target host name and target directory, choice of the copy program, and login name on a server. Parameters sent to a client depend on who the client is, what data it transfers, and on the server load. All parameters are configurable on the server side. The client can choose the copy program and its options, and can override the login name. In this schema, it's very easy to manage transfers from the server side. If a server goes down due to hardware failure or a scheduled maintenance, the system can be configured to avoid that server. When something else changes in the import environment, the change can be made transparent to the users. Thus, reducing down time

and the need for transfer/import outages. The web service is implemented using the SOAP protocol. Plain CGI is also supported.

Import jobs use the same import configuration. Thus, the whole system has a single global configuration file, managed by the administrators at SLAC.

KanTransfer: What This Tool Is

New transfer/import tool, called KanTransfer, is a set of client and server side applications written in Perl. It supports the transfer and import of both ROOT-based files and objectivity databases, as a set of files or tar archives. It has built-in support for bbcp and bbftp — two major WAN copy applications used in BaBar, optionally verifies file checksum, and can update BaBar's bookkeeping database after archiving the imported files.

FUTURE DEVELOPMENT

Load adaptive transfer

In order to use the network and server resources efficiently, the transfer parameters need to match the network and server conditions. Nowadays, wide area networks are capable of high speed transfers, but disk speed is not growing as fast. As a result, it doesn't make sense to accelerate network transfer if data cannot be written to import server's disk with the same speed.

There are a number of standard tools for network tuning, which probe the network between specified end points, and provide options, such as the number of TCP streams, or TCP buffer (window) size that provide the best performance. The problem, however, is that those tests are rather artificial, taking into account only network (i.e. testing memory to memory transfers), and not taking into account possible concurrent activity on the network and end

servers.

Our idea is to set such network options so that overall WAN copy performance is optimized, i.e. the criterion is high file transfer rate between given servers, disk to disk. Network options are supplied to a client's copy application and are not fixed. Instead, they are adjusted after transferring each file in order to "discover" the best combination, providing optimal transfer rate. Thus, transfer options are tailored to the *actual* load on the network and servers.

Reporting failures

Failures and problems are unavoidable, and are compounded by the peer-to-peer nature of the transfer. Most of the time, users do not properly report failures, nor do they mention all necessary details, possibly due to frustration caused by the failure. Sometimes, long email exchanges are necessary to find out what actually happened. Ideally, the developers and administrators should see *exactly* what a user sees on his screen.

Extending the idea of import as a web service, all the errors should be reported to the import server. KanTransfer is internally logging all the output to the screen, as well as many other parameters of the application and transfer, such as the version of KanTransfer, location where it was called from, command line options, configuration file options, timestamps, and all error and warning messages. When a failure occurs, KanTransfer attempts to send this information to the import side, if this is possible. Thus, the administrator has full information about the failure, as determined by the transfer tool, which greatly improves debugging and feedback.

SUMMARY

A simple tool has been created to facilitate BaBar's data transfer and import. The tool is in production since the end of 2003, running smoothly after usual startup difficulties. All BaBar data is archived with this tool, logging about 0.8 TB daily on average (see Fig. 3).
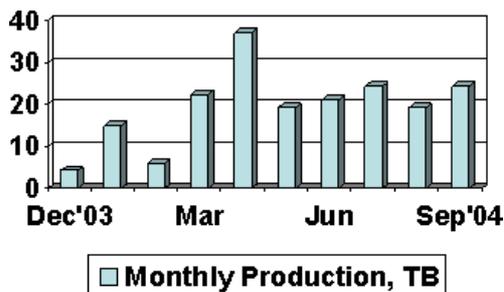

Figure 3: BaBar's monthly production of data in new Kanga format.

REFERENCES

[1] A. Hanushevsky, "The Next Generation Root File Server", CHEP'04, Interlaken, September 2004.

[2] http://www.objectivity.com

[3] J. Becla et al, "On the Verge of One Petabyte - the Story Behind the BaBar Database System", CHEP'03, San Diego, March 2003.

[4] Matthias Steinke et al, "How to build an event store - the new Kanga Event Store for BaBar", CHEP'04, Interlaken, September 2004.

[5] A. Hanushevsky, A. Trunov, L. Cottrell, "Peer-to-Peer Computing for Secure High Performance Data Copying", CHEP'01, Beijing, September 2001.