SLAC-PUB-7977
October 1998

The BABAR Prompt Reconstruction System

T. Glanzman, J. Bartelt, T.J. Pavel
Stanford Linear Accelerator Center
Stanford University, Stanford, CA 94309

S. Dasu
University of Wisconsin, Madison

For the BABAR Computing Group

Presented at Computing in High Energy Physics (CHEP å98), 8/31/98ã9/4/98,
Chicago, IL, USA

Stanford Linear Accelerator Center, Stanford University, Stanford, CA 94309

Work supported by Department of Energy contract DE AC03 76SF00515.

SLAC-PUB-7977
October 1998

The *BABAR* Prompt Reconstruction System[1]

T.Glanzman, J.Bartelt, T.J.Pavel
Stanford Linear Accelerator Center,
Stanford University, Stanford, CA, 94309

S.Dasu
University of Wisconsin, Madison

For the BABAR Computing Group

1. *BABAR* Overview

BABAR is an experiment designed to explore the nature of CP violation and other physics in the *B B*-bar system starting in the Spring of 1999[1]. The experiment is situated at the Stanford Linear Accelerator Center where the former PEP electron-positron storage ring has been upgraded to an e^+ e^- asymmetric storage facility[2]. The electrons at 9 GeV/c and positrons at 3.1 GeV/c will collide at the Y(4S) with a luminosity of 3 x 10^{33} cm^{-2} s^{-1}. The *BABAR* detector will record the B and B-bar decays resulting from the Y(4S). With careful vertexing, particle ID and other measurements, asymmetries between the B and B-bar decays will be analyzed.

The expected data rate reaches 100 Hz of ~32 kByte events recorded to mass storage. This results in ~10^9 logged events per year of operation. The raw data, combined with reconstructed and simulated data, is expected to yield ~300 TB of stored data per year. With this large flow of data, it becomes essential to optimize the automatation and reliability associated with the initial phases of data processing.

2. Motivation & Requirements

The software associated with *BABAR*'s data processing began development simultaneously with the detector hardware systems. Historically, software development has frequently lagged behind the hardware and thereby caused delays in publication quality analyses. Inparticular, it was noted that careful calibrations and alignments often took months to perfect. There was, of course, a strong desire to analyze data quickly not only for physics output, but also for feedback to the detector operations staff so they could sense and fix problems when they arose. This was deemed particularly important during the startup phase of the experiment. Thus, two years ago, the collaboration mandated a "prompt reconstruction" system be designed, put in place and ready for the first day of data taking in 1999. The basic requirements included the following:

- Complete reconstruction for 100% of all events
- Event logging into the Objectivity event store
- Detailed quality monitoring data for feedback to detector operators, thus
- Low latency (nominally within 2 hours of data acquisition)

1.This work was supported by Department of Energy contracts DE-AC03-76SF00515. (SLAC) and DE-FG02-95ER40896 (University of Wisconsin).

- Event tagging based upon analysis of fully reconstructed events
- Dynamic calculation of traditionally “offline” constants to be used in subsequent reconstruction jobs
- Dynamic calculation of alignment corrections

As a consequence of these requirements, it was decided to create a new Prompt Reconstruction component within the *BABAR* Online System. As such, the Prompt Reconstruction is subject to many of the rigorous requirements of the Online System.

- System must be completely automated and operable within the Online System
- Quality monitoring results must integrate into “end run” summary system
- Alarms must be recorded and handled properly
- System must be fault tolerant
- Perhaps most importantly, the Prompt Reconstruction system must incur no deadtime

3. Hardware Platform

The basic hardware platform with which to make the system work is illustrated in Figure 1. It

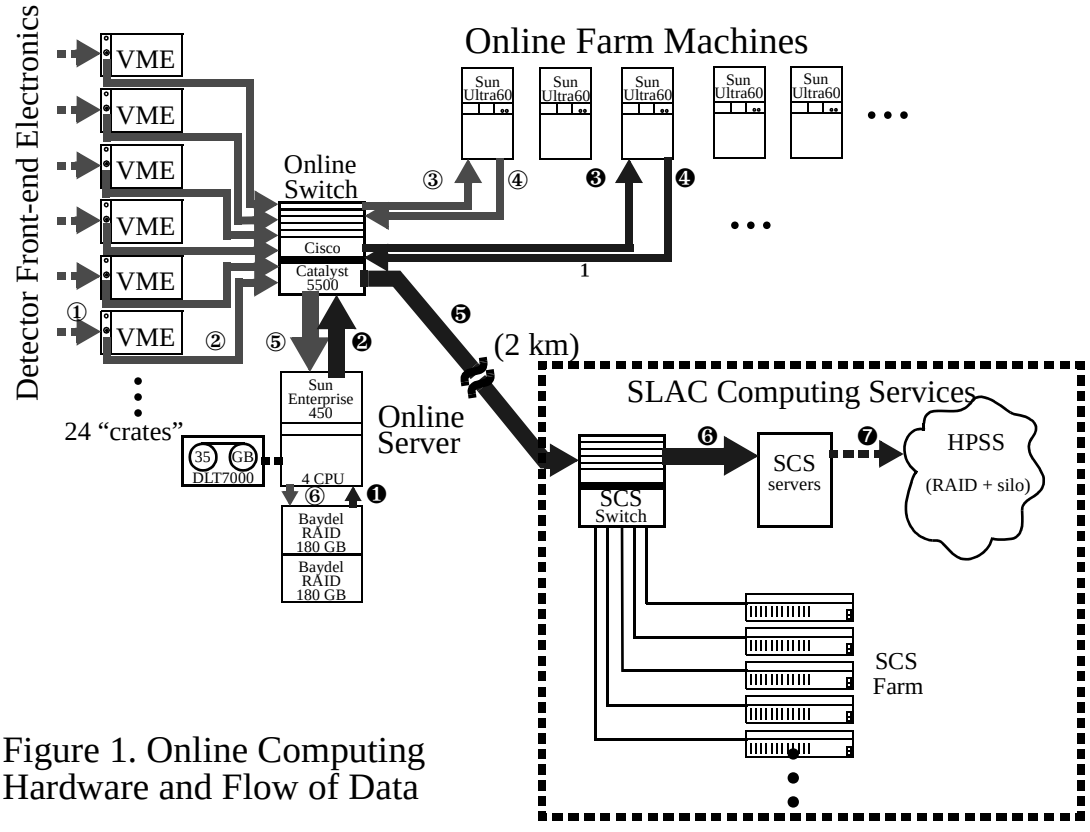

Figure 1. Online Computing Hardware and Flow of Data

consists of Unix workstations[3], VME processors, standard networking equipment[4] and mass storage. Logically, the Prompt Reconstruction system must process all data selected by the final level of trigger and deposit the results into the event store. Thus, a system was designed where the flow of data through the system is indicated in the figure by the sequence of white numbers in black circles.

The size of the Online Farm is likely to contain many tens of machines. While a smaller number of large SMPs would be possible, it is not currently the most cost effective solution. Each machine must run its own copy of the operating system and be configured for the network. When new machines appear or OS upgrades (e.g. security patches) require installation, each machine must typically be manipulated independently. Because the *BABAR* Online System is currently dependent on the Solaris operating system, we have investigated some vendor-specific tools to ease the management of large numbers of machines[5]. The particular tools used include: Admin-Suite; Auto-client; and, CacheFS.

With these tools, one can bring up a new machine or apply OS patches/upgrades to a large number of machines in minutes, effectively turning an individual machine into a field-replaceable unit. This is accomplished by first configuring an OS Server machine (this takes more than minutes) and then cloning a prototype `root` file system for each of the client machines. The OS Server thus retains client-specific `root` partitions and a single shared copy of the `usr` file system. This is complemented on each client machine by local copies of these directories in the CacheFS. See Figure 2. Thus, the first time a new client machine is booted its local cache is ini-

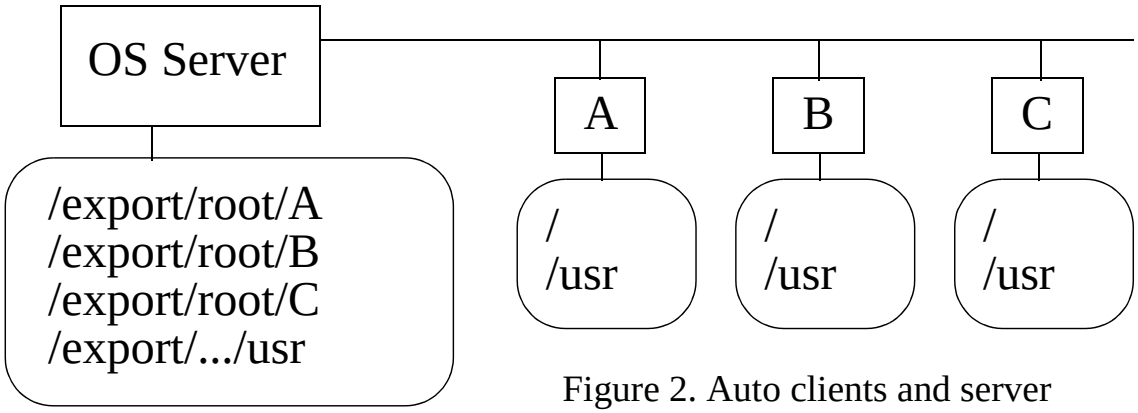

Figure 2. Auto clients and server

tialized so that on subsequent reboots only local disk access is required for most operations. There are tools, for example, to apply OS patches to all of the the client-specific /root partitions, then requiring each client to refresh its cache. In our tests, the time to configure a new Sun Ultra5 machine dropped from ~400 minutes to ~10 minutes, complete. While there are some minor problems with the scheme, it is very promising and will be adopted for the Online Farm machines.

4. Software Development

Starting in the Summer of 1996 when the Prompt Reconstruction effort was commissioned, the design was already constrained by an existing set of "*BABAR* standard" tools. The most important of these included the use of the Unix operating system, the Objectivity database[6] and the C++ programming language (to be used in a proper objected oriented manner). Within *BABAR*, a set of standard tools was also in place (or promised), such as a framework within which to perform a generic calibration, a distributed histogrammer, a code management environment and a standard execution framework. Still, the envisioned multi-process and performance-sensitive Prompt Reconstruction system required additional tools. After some accelerated R&D it was decided to use the ACE wrappers[7] (a package of C++ wrappers around low-level real-time functions and a collection of high-level design patterns) for performance-sensitive data transport, CORBA for interprocess control communications; and Java (with CORBA) to build GUIs.

Finally, all members of the core Prompt Reconstruction development team took two one-week courses in object oriented analysis and design from Object Mentors, Inc[8]. The ROSE CASE tool was also used to a limited extent, although mostly to draw various types of UML diagrams.

5. The Software

The software design not only reflects the requirements as outlined above, but also folds in certain decisions about the operation of the system. During data acquisition, for example, single events are first built in one several Unix workstations (constituting a "farm") where the Level 3 trigger is subsequently performed. After a Level 3 accept, events are funneled into a single intermediate store on disk. Every 30 minutes, this file of events is closed an a new one opened. Once closed, such a file is eligible for processing in the Prompt Reconstructin system. The flow of data is indicated in Figure 1, in which each step of the data acquisition and the subsequent Prompt Reconstruction is indicated by the numbered arrows This scheme was chosen to improve the reliability of the entire system by virtually eliminating the possibility that Prompt Reconstruction could become a source of experiment deadtime. That said, the following list of six process types constitutes the Prompt Reconstruction system.

1 **Global Farm Manager (GFM)** - manages the entire farm of CPU resources; manages event blocks from OEP and to Prompt Reconstruction; also manages each instance of Prompt Reconstruction
2 **Global Farm Daemon (GFD)** - lightweight "ping" process running on all (potential) CPUs in the farm
3 **Prompt Reconstruction Manager (PRM)** - manages one instance of Prompt Reconstruction
4 **Logging Manager (LM)** - directly handles the event stream from OEP to disk and from disk to Prompt Reconstruction[9]
5 **Prompt Reconstruction Daemon (PRD)** - directly handles incoming event I/O from the LM for the PRF (this isolates the I/O and event bookkeeping from the PRF)
6 **Prompt Reconstruction Framework (PRF)** - standard BABAR execution framework along with the Reconstruction, Calibration/Alignment, Monitoring and other user code. From the perspective of the Online system, this is "unreliable" code!

The relationship of these processes to each other along with the various communication mechanisms used are shown in Figure 3 In this figure are indicated one instance of the Online Event Processing (OEP) and two independent instances of a Prompt Reconstruction. A "reliable server" simply means one that has extra reliability features, such as redundant power supplies, ethernet adaptors, etc. The Logging Manager is, by virtue of being an element in the data stream, a particularly complex item; it is described in more detail in another paper at this conference. Note that each processing node receives a PRD/PRF pair. The PRD has the extra responsibility of monitoring the health of the PRF and taking action if it becomes ill or dies. It is imagined that in the steady state only a single instance of Prompt Reconstruction will be running at any given time, but additional instances could be used, for example, to offer alternative ways of processing a data

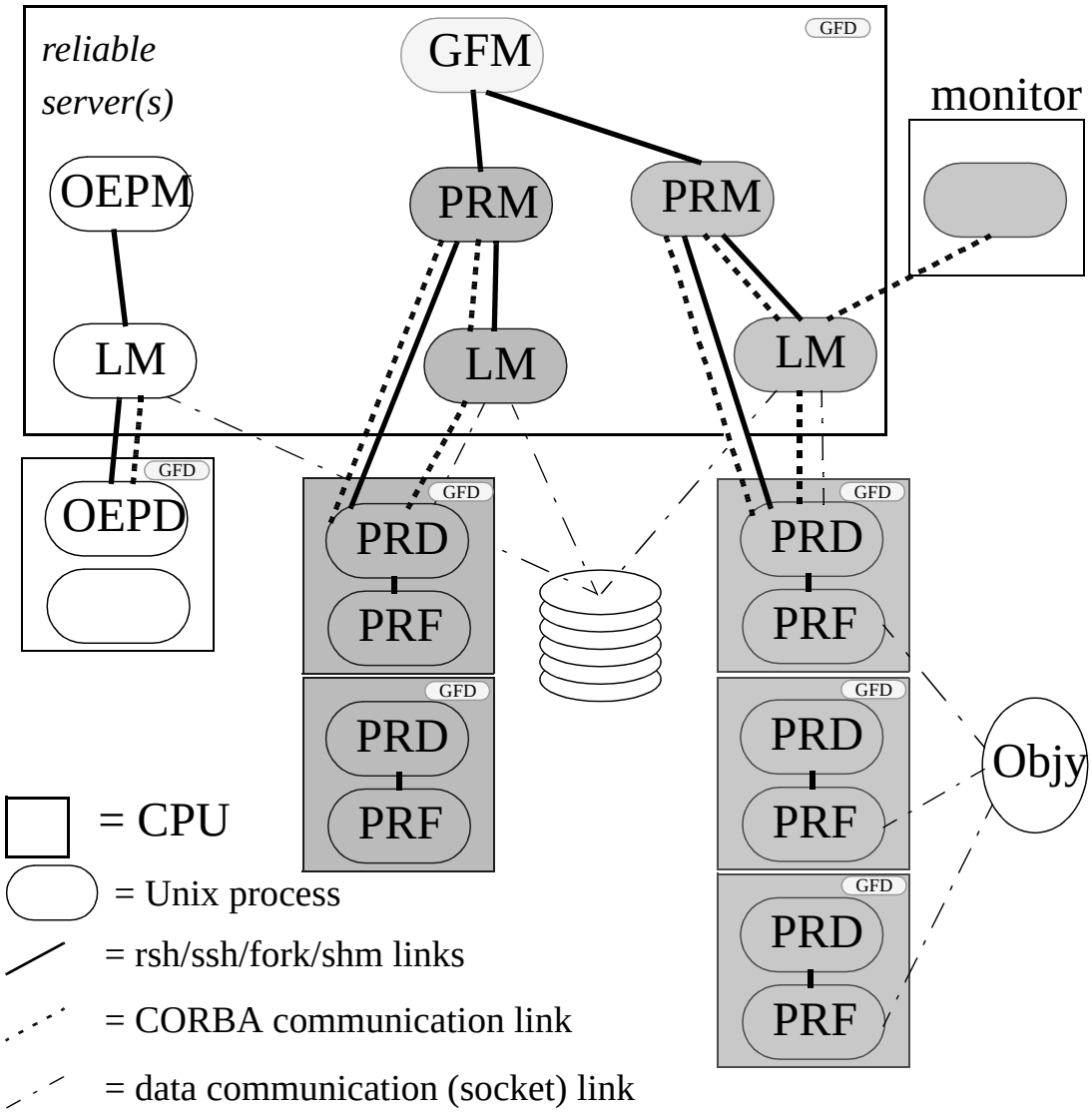

Figure 3. Cooperating Prompt Reconstruction Processes

backlog. .

6. Constants and Quality Monitoring

Many constants, such as electronics channel pedestals and gains, are generated during special calibration runs of the detector. Others, such as the drift chamber time-to-distance relationship and corrections to the relative alignment of the drift chamber and silicon vertex detector, require a large number of completely reconstructed events. These latter calculations are done within the Prompt Reconstruction and are referred to as “datastream calibrations”. Constants from datastream calibrations are based upon one or more 30 minute blocks of data.

One novelty of this approach is that constants generated from data block N will then be used in the generation of constants from data block N+1. This eliminates the need for an iterative process

in which the same event may have to be processed mulitple times for the purpose of generating constants.

Complications arise from two other sources: the need to collect and combine running sums (or other ingredients to the constants generating code) from the many computers participating in Prompt Reconstruction; and, the need to integrate data across many 30 minute data blocks for certain types of calculations. These issues have been addressed by the development and use of two transient Objectivity databases, the Spatial database - for collecting and combining data across many machines for one slice of time, and the Temporal database - for integrating data across many data blocks[10].

Quality Monitoring is accomplished via a distributed histogrammer system[11]. Using a generic histogramming interface, this system is able to harvest and combine sets of histograms from multiple processes and, in the case of Prompt Reconstruction, store the result into an Objectivity database.

7. Summary and Status

A Prompt Reconstruction system has been described that will quickly and reliably perform a first pass reconstruction on 100% of the data from the BABAR experiment. This system will also provide for the calculation of constants and acquisition of monitoring histograms for the running detector. Approximately 70% of the system is beyond the final design stage and much of that has been implemented. A fully functional system is expected by the BABAR Cosmic Ray run in November 1998, and the complete system by April 1999.

8. Acknowledgements

We would like to acknowledge and thank a number of former members of the Prompt Reconstruction team who provided important contributions to the R&D, design and/or implementation of the system. Colin McCreight, now an undergraduate at Princeton University, helped to develop, run and analyze CPU and network performance tests. Sasha Telnov, a graduate student at U.C. Berkeley helped to investigate and put into production the Solaris auto-client system. Ryan White, now an undergraduate at Caltech, built a CORBA-based monitor for the system and provided the first implementation of the PRD. Stephane Bonneaud, an undergraduate in University Louis Pasteur, Strasbourg, France, generously donated his time and developed our first Java/CORBA GUI.

9. References

1 *BABAR Technical Design Repor*t, SLAC-R-95-457, 1995 (also online at http://www.slac.stanford.edu/BFROOT/doc/TDR/)
2 *An Asymmetric B Factory Based on PEP, Conceptual Design Report*, SLAC-372, 1991
3 *Choosing CPUs in an Open Market: System Performance Testing for the BaBar Online Farm*, T.J.Pavel, CHEP paper 31, http://www.hep.net/chep98/
4 *Network Performance Testing for the BaBar Event Builder,* T.J.Pavel, et al, CHEP paper 31, http://www.hep.net/chep98/
5 *Management of Computer Farms at BABAR*, *BABAR* note 446, Alexandre V. Telnov

6 *First Experience with the BABAR Event Store*, David Quarrie, CHEP98 paper 33, http://www.hep.net/chep98/
7 Information on ACE and TAO, plus the code can be found here: http://www.cs.wustl.edu/~schmidt/
8 Information about Object Mentors, Inc. can be found here: http://www.oma.com/
9 *Event Logging and Distribution for the BABAR Online System*, Sridhara Dasu, et al, CHEP98 peper 74, http://www.hep.net/chep98/
10 *Databases for BABAR Datastream Calibrations and Prompt Reconstruction Processes*, John Bartelt, et al, CHEP98 paper 47, http://www.hep.net/chep98/
11 *Distributed Histogrammer*, Scott Metzler, et al, CHEP98 paper 62, http://www.hep.net/chep98/