The BABAR Prompt Reconstruction System, or *Getting the Results out Fast*: an evaluation of nine months experience operating a near real-time bulk data production system *

T.Glanzman[1], *J.Bartelt*[1], *R.F.Cowan*[2], *S.Dasu*[3], *G.Grosdidier*[4], *F.di Lodovico*[5] and *F.Safai Tehrani*[6] for the *BABAR* Computing Group

[1] Stanford Linear Accelerator Center, Menlo Park, CA, USA
[2] Laboratory for Nuclear Science, M.I.T., Cambridge, MA, USA
[3] University of Wisconsin, Madison, WI, USA
[4] LAL-IN2P3-CNRS, Université Paris-Sud, Orsay, France
[5] University of Edinburgh, Scotland, UK
[6] INFN, Sezione di RomaI, Rome, Italy

Abstract

The *BABAR* experiment at SLAC has been operational since May 1999. An ambitious program to completely reconstruct 100% of all physics events within two hours of their acquisition was launched. In addition, this system was intended to provide a nearly continuous "rolling calibration" and extensive detector monitoring for feedback into the working experiment. We succeeded in processing the very first PEP-II collisions recorded by the detector within a few hours. Unfortunately, problems with code reliability, computing and network infrastructure, Objectivity and operational efficiency prevented us from maintaining this short latency. Increasing accelerator luminosity and the cancellation of a much-needed scheduled down period complicated our ability to upgrade the system. After months of struggling with code release policies, hardware upgrades, and extensive Objectivity development, we are now within sight of our primary performance goals.

The focus of this paper is to summarize the more important steps required to make this project a success with emphasis on lessons learned. Overall performance, current status of the running system and future plans will also be presented.

Keywords: production analysis, reconstruction, automation, database, BABAR, Objectivity

1 Overview

BABAR is an experiment to explore the nature of CP violation and other physics in the $B\overline{B}$ system [1]. The experiment is situated at the Stanford Linear Accelerator Center and records collisions produced at the PEP-II collider, an e+ e- asymmetric storage ring [2]. The electrons at 9 GeV/c and positrons at 3.1 GeV/c collide at the Y(4S) at a soon-to-be-achieved design luminosity of 3×10^{33} cm^{-2} s^{-1}. The *BABAR* detector records the B and $\overline{B}$ decays resulting from the Y(4S). With careful vertexing, particle ID and other measurements, asymmetries between the B and $\overline{B}$ decays are being analyzed. Data collection began in the Spring of 1999.

The design data rate is 100 Hz of ~32 kilobyte 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 yields ~300 TB of stored data per year. Given the need for timely data quality checks and a desire to produce publication quality results quickly, it was essential to optimize the automation and reliability of the data processing system. This paper describes our experiences toward making this system successful. The overall design of the Prompt Reconstruction system has been described in a previous paper [3] while several of its

*.This work was supported in part by U.S. Department of Energy contracts DE-AC03-76SF00515 (SLAC), DE-FG02-95ER40896 (University of Wisconsin) and DE-FC02-94ER40818 (M.I.T.).

component parts are described in other papers at this conference [4][5][6].

2 The System

The basic operation of Prompt Reconstruction consists of running a sequence of jobs, each of which processes the events from one "run", typically 30-40 minutes of data (the time between beam fills in PEP-II), corresponding to 180k to 240k events. In order to provide timely feedback to the operating detector, these events are processed in parallel on 100-200 machines, currently Sun Ultra5 CPUs running Solaris 2.6. Because of a novel method of calculating detector constants, we must adhere to a strict ordering of this processing.

As of today the entire system consists of 100 "farm" machines which perform the reconstruction, constants-generating and monitoring, plus five servers. One server reads the raw data and distributes single events to the farm nodes for processing and maintains bookkeeping. The remaining servers are for support of Objectivity. We currently have two 4-cpu machines dedicated to the event data being written to Objectivity, one lock server and one journal server.

Prompt Reconstruction is part of the *BABAR* online system. As such, we also have certain performance requirements to meet, including both throughput and latency. The basic requirements are to provide sufficient processing throughput to keep up with the 100 Hz incoming event rate, and to complete that processing within two hours of acquisition. The latency requirement is important both for providing timely feedback to the detector operations crew, but also simply to make fresh data available for physics analysis as soon as possible.

3 Getting Started

The first data was recorded by *BABAR* on 26 May 1999. However, long before that occurred, many cosmic ray events were recorded starting in March of that year. The excitement and urgency that accompanies the start up of any large project required that the Prompt Reconstruction team take on a major operations responsibility in addition to our development. This included involvement in such activities as coordinating the installation and configuration of the machines to be used for processing, coordinating the right code releases to be used, coordinating the constants stored in the database by the various detector subsystems and, of course, processing all new data.

Early experience processing the first cosmic data was mixed. We rapidly discovered an initial set of problems. Both the Objectivity and the *BABAR* codes proved problematic. We discovered that scaling the number of machines to about 20 yielded expected results, but performance plateaued and even decreased beyond that. A key issue to be resolved before this system could hope to process at 100 Hz was the Objectivity scaling problem. Additional details about the role of Objectivity in this story are presented in two other talks at this conference [7][8].

By 26 May 1999, or "First Data", we had assembled a system which seemed to work reliably on cosmic ray data, although not with the needed capacity. Almost immediately we encountered a new set of problems. Because the reconstruction code was in rapid development at that time, updating to the latest revision resulted in many crashes. Problems often did not surface during limited testing by the code developers, and only appeared in the context of production. We also learned that the pressures brought to bear on the Objectivity servers were sensitive to the differences between cosmic ray and colliding beam data (e.g. processing rates, hence DB writing rates were different, the number of persistent objects increased due to the events being more complex, etc.) It also became clear that we needed to develop tools to monitor a growing number of widely distributed system and process-level quantities just to understand what was happening [9]. Finally, we discovered how demanding a 7 x 24 round-the-clock operations role could be for a small development group. At this point, all but essential development halted just to perform operational duties. Although we had managed to process the First Data within a few hours of its acquisition, the situation was dire.

4 The Onset of Real Data and Growing Pains

During the course of running Prompt Reconstruction a large variety of problems was encountered. Many of these problems might have been prevented if the experiment had had less ambitious schedule goals. Thus, time still needed for development, installation and testing merged into the need for operating the system. Some of the more important complications encountered over the past nine months are briefly described below.

- **Commissioning a lot of new code**. With $O(10^6)$ lines of code, some of it exercised well, some of it brand new, reliability was not an option. Code crashes were routine and slowed down the processing rate significantly. Shortly after First Data, a new group was assembled consisting of code experts from each of the detector subsystems, Prompt Reconstruction and computing management. The group met twice weekly to discuss (and approve) any proposed changes to the code or constants. The BABAR Computing Coordinator, after discussion, made executive decisions about each proposed change. To the code developers this must have seemed a heavy-handed way to manage the software, but to operations, it made the difference between running and not running.
- **Unstable infrastructure**. In many cases the computers and network components used by OPR were very new, having been installed only during the previous few months -- using the "just in time" approach to purchasing/installing computing equipment. Heroic efforts within the SLAC Computing Services allowed this new equipment to be installed and configured in record time. However, there were a number of new technologies being used, e.g. gigabit ethernet, a new model of switch, and the Sun Autoclient system for system management of 100 machines. There were occasional glitches and meltdowns in these systems. We also depend upon parts of the data acquisition infrastructure at the detector which also had its moments of instability. Finally, the power systems, including UPS, failed numerous times causing isolated or widespread outages. The passage of time and upgrades has helped alleviate these problems.
- **Core OPR code was not yet complete**. The basic mechanisms for submitting jobs worked well on day one. However, automating the submission, monitoring and checking completion status for a sequence of jobs was not yet in place. Over the summer of 1999 such a system was commissioned. In November of 1999, another important piece, the Global Farm Manager, was commissioned [5].
- **Fragility of Objectivity**. On the server side, we discovered misconfigurations and bugs in the code. Due to dealing with a third party software vendor, bug fixes were sometimes not possible to schedule in a timely manner and a great deal of work went into writing code workarounds. In general, in the early days, so little was known about the internal workings of the Objectivity code that seemingly small changes to our code caused large perturbations in system behavior. Further, there exists almost no activity logging capability within the Objectivity software. Time spent in battle conditions eventually helped us learn the location of many traps and pitfalls, and code was modified to avoid them.
- **Those dreaded locks**. The manner in which *BABAR* originally coded to the Objectivity API made generous use of locks and was not particularly well-suited for large numbers of machines running simultaneously. This resulted in various types of lock conflicts, long delays awaiting for locks to be released and even deadlocks. Also, when individual machines crashed locks would typically be left behind -- to interfere with other jobs. In most cases, these problems were laboriously eliminated by fixing the code or changing the code to work-around obscure limitations within Objectivity.
- **Bottlenecks in the Objectivity system**. The volume of data being sent to Objectivity and the detailed manner in which it is done (e.g. size of individual objects, their placement within the federation, the use of locks, etc.) rapidly demonstrated that we had seriously underestimated the capacity of the system on the server side. The resultant bottlenecks prevented the processing rate from increasing linearly with the number of processing nodes. Much of the database-related work over the past months has focused specifically on how to scale the system.
- **Database "sweeps"**. Due to access limitations, lock problems, and fragility of Objectivity, a scheme was developed where each major phase of the data processing chain was assigned its own Database Federation. Due to limitations in the Objectivity code, a single user application can, at most, access a single Fed-

eration. Therefore, processed data written to the OPR federation must be frequently transferred or "swept" into the analysis federation. Similarly, online constants generated at the experiment must be swept into the OPR federation prior to processing the corresponding data. Regular scheduled outages accommodate these sweeps, but they directly impact the OPR throughput.

- **Rolling calibrations and ordering of processing**. Producing constants at the end of a processing instance requires collecting partial statistics from all participating machines during a special end-of-run sequence. A consequence of the "rolling calibration" concept is that data runs must be processed in order. But runs which had upstream DAQ problems were routinely skipped, and various run-skipping policies were ineffect at different times to help keep up with the flood of data. Also, as this code was barely out of development and proved a further strain on Objectivity, it was disabled at an early stage. A further complication resulted from having multiple federations in that electronic and OPR calibrations originated in different federations and had to be carefully combined rather than blindly "swept" from place to place. Rolling calibrations are only now being recommissioned.
- **Large number (and fragility) of code releases over time**. Between the period of 18 June 1999 and 6 January 2000, the total number of code releases for the main event reconstruction, calibration and monitoring was 24. This is nearly one new release per week on average. Each time a new release is introduced, a significant amount of time is spent flushing out problems that occur only with large numbers of events or machines. It has been a long-standing goal to seriously reduce the rate of new releases, but that is understandably a very difficult policy to accomplish.
- **Labor to supply extensive testing**. Because of limited resources, it was not possible to outfit each developer within *BABAR* his/her own Prompt Reconstruction system. Also, running a large, distributed processing system takes a bit of training. Thus, our team was obligated to assist the different contributing software development groups with running tests of their code. Often these tests were repeated many times and exercised new parts of the system. The result was identifying a special person specifically to help with testing.
- **Culture surrounding the onset of operations**. The general frenzy associated with First Data led to much frustration when the processing system stumbled. This resulted in a heavy administrative cost to the core development group in attending special "emergency councils" and preparing long explanatory presentations at collaboration and other meetings.
- **Need to *reprocess***. Because the reconstruction code was (and is) continually changing, people analyzing the data were frustrated not to have a nice block of data all processed identically. Therefore, requests for reprocessing began to arrive. As a result, a second compute farm and a whole new operations team is now being assembled to deal with this.
- **Event filters**. Non-physics (beam-gas or other background) and Bhabha/two-photon events need careful filtering to minimize the CPU time spent on such events. Any type of filter is controversial and the thresholds for selecting such events are continually changing. We now have in place such filters and, to save time, refrain from writing the raw data for such events into the database. Very roughly 25% of all events are currently passed through the filters, fully processed and stored in the database.
- **Happily, a large amount of data to process**. The luminosity of PEP-II has exceeded expectations, blessing us with a large amount of data. So there is less time for testing and contemplation and more pressure for production running. One significant scheduled down time in August was cancelled, eliminating a hoped-for period during which development could resume. This problem was solved by the creation of a new OPR Operations team consisting of volunteers from within the collaboration.

In spite of many frustrating shifts, the system has been, with minor exceptions, kept running 7 x 24 (24 hours per day, 7 days per week) since late May of 1999.

5 Summary of Achievements

In spite of the aforementioned travails, the system performed surprisingly well. The results of processing for the period 26 May 1999 through 14 January 2000 (33 weeks + 2 days) are as follows.

Total events processed: 250M (12.4 Hz averaged over the entire 33 week period)
Total events processed as "final": 179M
Total colliding beam raw events acquired: 194M
Current steady-state processing rate: 55 Hz using 100 machines
Average processing rate over select 11-day period: 26 Hz
Processing latency (run creation to first processing attempt):
    Total runs processed: 3178
    First Data processed within hours of its acquisition
    Total runs processed within 2 hours: 7 (0.2%)
    Total runs processed within 2 days: 858 (27%)
    Total runs processed within 2 weeks: 2268 (71%)

6 Left Undone

There is still core development to complete, notably a component called the Global Farm Manager (GFM) which handles overall machine and processing management, along with bookkeeping. These functions are currently being done by script or by hand. The GFM will also participate in overall monitoring of the full system and act as an early alarm sentry when something goes wrong. Our current script-based job submission system will be replaced with C++ [4].

Final hardware allowing us to scale up the system to accommodate an input rate of 100 Hz is just now coming online. This is scheduled to be in production by 15 March 2000. However, as the PEP-II luminosity increases and either the 100 Hz trigger rate is allowed to increase or the fraction of interesting physics events increases, we may discover our system, once again, on the edge of needed performance.

7 Conclusions

The *BABAR* Prompt Reconstruction system has been commissioned and in continuous production since May of 1999. During that period it has successfully processed essentially all interesting collision events recorded by *BABAR* during that period, sometimes more than once. Numerous problems prevented us from reaching our original goal of keeping up with data in near-real-time, but we expect to meet that goal within the next two months.

Our design for a system as complex as Prompt Reconstruction was necessarily based upon various premises, some of which turned out to be invalid. We (re)learned that a substantial amount of the engineering design is best accomplished *after* the system begins to function. Prototyping and even building a first version of certain components using scripting languages, which much more quickly adapt to changing circumstances than compiled code, proved to be a success.

The luxury of performing code development and testing is quickly lost once data begins to appear. It is important to have an operations group in place *before it is needed* so that they are properly trained and able to keep the load off of the developers until the project is fully commissioned.

Large codes, such as event reconstruction code, cannot be allowed to change too quickly or for little reason. For a production system to run dependably, strict change control and careful management are absolute requirements.

The *BABAR* computing management and, in particular, the Computing Coordinator, Terry Schalk, was very sensitive to the problems encountered and responsive when additional hardware or people were needed. Without such support, the level of success we demonstrated would not have been possible.

8 References

1 *BABAR Technical Design Repor*t, SLAC-R-95-457, 1995 (also online at http://www.slac.stanford.edu/BFROOT/www/doc/TDR/)

2 *An Asymmetric B Factory Based on PEP, Conceptual Design Report*, SLAC-372, 1991

3 *The BABAR Prompt Reconstruction System*, T.Glanzman, et al, CHEP98 paper 52, http://www.hep.net/chep98/PDF/52.pdf

4 *The BaBar Prompt Reconstruction Manager: a real-life example of a constructive approach to software development*, Francesco Safai Tehrani, et al, CHEP2000 paper 180

5 *Sending Commands and Managing Processes across the BABAR OPR Unix Farm through C++ and CORBA*, G.Grosdidier, et al, CHEP2000 paper 161

6 *Event Logging and Distribution for the BaBar Online System*, S.Dasu, et al, CHEP2000 paper 138

7 *Operational Experience with the BaBar Database*, D.Quarrie, CHEP2000 paper 103

8 *Improving Performance of Object Oriented Databases, BaBar Case Studies*, J.Becla, CHEP2000 paper 110

9 *Visualization Tools for Monitoring and Evaluation of Distributed Computing Systems*, R. Cowan, et al, CHEP2000 paper 186.