Event Logging and Distribution for the BaBar Online System

S. Dasu[1], *T. Glanzman*[2], *T. J. Pavel*[2†]
For the BaBar Prompt Reconstruction and Computing Groups

[1] Department of Physics[‡] , University of Wisconsin, Madison, WI 53706
[2] Stanford Linear Accelerator Center, Stanford, CA 94309. [§]

Abstract

We discuss the BABAR experiment software and computing infrastructure used for event logging and distribution to prompt reconstruction system. The raw data for typical BABAR events is about 30-50 kB per event. During the data taking a 100 Hz stream of events is collected from a farm of 32 online computers amounting to about 3-5 MB/s. A robust and efficient multi-threaded, TCP/IP based logging manager software was written to serve this purpose. Monitoring of the status of the program is achieved using CORBA. The same program is also used to distribute the logged events to a larger farm of 200 unix processors to process data promptly. This prompt processing involves full reconstruction and calibration programs before final storage in the Objectivity database. We describe the performance of underlying hardware and software, and address scalability of the program.

Keywords: TCP/IP, CORBA, Network, Computer Farm, Logging

Introduction

The BABAR experiment [1] at the Stanford Linear Accelerator Center is built to study the particle and anti-particle asymmetries. These small asymmetries due to a phenomenon called the CP violation, are hitherto observed only in the K mesons are also expected to be observable in the B mesons as well. Both validation of the Standard Model explanation for the CP violation phenomenon and searches for beyond the Standard Model explanations require the CP violation data in the B meson system. The observation and study of the CP violation phenomenon in the B meson system requires production of at least 10^9 electron-positron collision events per year at the Υ(4S - 10.58 GeV) resonance. These events are measured by the BABAR experiment with high precision, using its silicon vertex detectors, drift chamber, ring imaging Cherenkov detector, CsI crystal calorimeter and resistive plate chambers. The raw data for typical events from these sub-detectors adds up to about 30-50 kB per event. During the data taking the collected stream of data from the detector is expected to run at about 3-5 MB/s. A robust and efficient computing system is required to receive this data-flow and reconstruct the events before their final storage, with raw and reconstructed data, in an object database. These data, accumulated over years, are analyzed by physicists to study the B meson physics.

The data from the BABAR sub-detectors are collected by custom VME electronics and presented to the data acquisition system using a set of generic read out modules via VME based PowerPC processors. These processors running VxWorks operating system communicate outside the crate using Fast Ethernet interfaces. This data-flow system and its software are described in

[†] On leave from Stanford Linear Accelerator Center.
[‡] This work was supported in part by Department of Energy contract DE–FG02–95ER40896.
[§] This work was supported in part by Department of Energy contract DE–AC03–76SF00515.

detail elsewhere [2]. The data from various subsystem processors are assembled into a full event on a farm of Unix computers. These data arriving at a combined 2 kHz rate into the farm are reduced to a 100 Hz flow by online event processing framework [3] that supports level-3 trigger software [4]. These 100 Hz events are archived for subsequent detailed analysis.

Although each event is independent of the others, the environmental information, e.g., for calibrating the drift time in the chambers, requires events contiguous in time. Therefore, it is important to collect these streams of events from various computers into a single stream to make collections of events spanning about an hour. These event collections for each run are fully reconstructed to determine calibration constants for subsequent data processing. The "prompt reconstruction" of these events requires running large and complex software programs [5]. It provides physicist early access to the fully reconstructed events enabling timely publication of results. Although the reconstruction is expected to be prompt, a latency of few hours has to be tolerated. Therefore it is necessary to provide a data buffer to separate these large programs from the dataflow software. The size of these one hour data sets is several giga-bytes strongly suggesting disk file buffers. These raw data files are also archived in the HPSS mass storage system so that later reprocessing of the data can also be done. In this paper we describe the software, the logging manager, that collects the events from several Unix computers to make these "intermediate" event files and distributes those events to the processor computers for reconstruction and final archival.

System Implementation

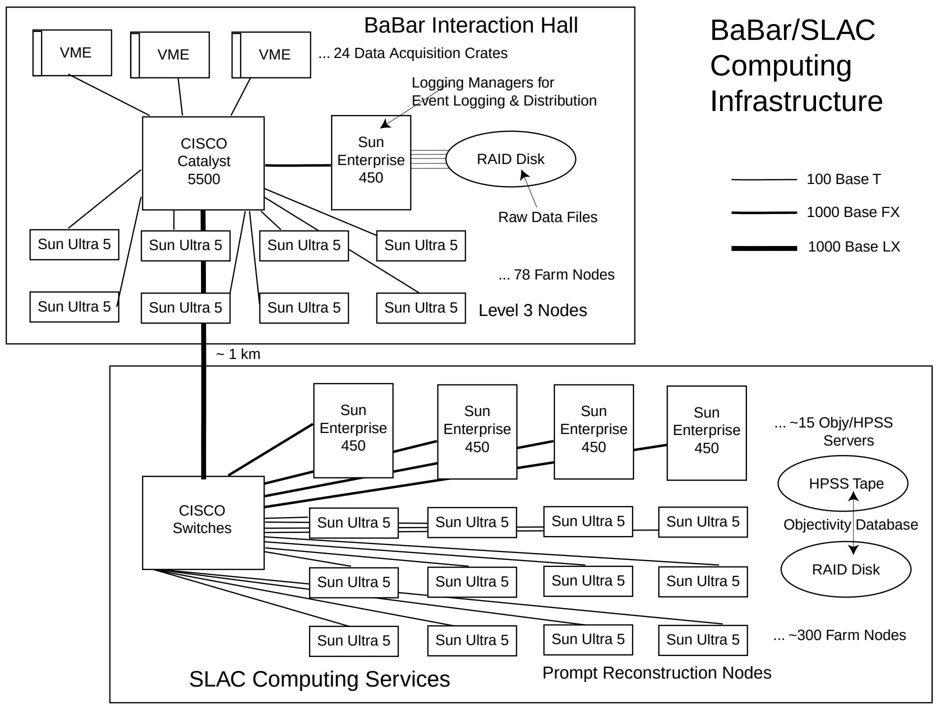

Figure 1: Diagram of the BaBar /SLAC online and prompt reconstruction computer farms showing the VME PowerPC front-end computers, the online farm of Sun UltraSPARC computers, the Sun Enterprise 450 server with the intermediate store disk array and the SLAC computer center database servers and worker nodes, all connected using Cisco Catalyst 5500 switches.

The hardware platform for the BABAR /SLAC computer farms is illustrated in Figure 1. The VME computers and the individual farm nodes have Fast Ethernet (100Base-T) connections to the Cisco Catalyst 5500 switch. The Online server computer, where the logging manager runs, supports an array of high performance RAID disks for intermediate file storage, and is connected to the online switch using two Gigabit Ethernet (1000Base-SX) links. One of the two network interfaces is reserved for communications between VME processors and the server and the other is used for all other communication including event logging and distribution. A separate farm of Unix computers in the SLAC Computing Services department support the BABAR experiment for running both reconstruction software and final data archival in a HPSS based Objectivity database. The SCS farm is connected to the BABAR online switch using a Gigabit Ethernet (1000Base-LX) link. The choice of computers and IP interconnects are based on careful evaluation of the commercial technology [6] considering both performance and cost issues.

Several Unix processes and their IP or shared memory based communication links involved in orchestrating the online platform to enable the intermediate file storage and data processing before the final storage in Objectivity database. The reliable server processes run on the online server computer and command the daemon processes on the rest of the farm to perform specific tasks, e.g., level-3 trigger or prompt reconstruction (PR). The logging manager processes collect or distribute the data using TCP/IP sockets for high performance, whereas the monitor and control data is exchanged using CORBA. When the processes are co-resident in a memory space, e.g., PR daemon and PR framework, shared memory is used to make the best use of the system resources. The details of the prompt reconstruction [5] are described elsewhere. Here, we report only on the Logging Manager process.

The software is built using object oriented design. The development phase involved modeling the objects using Unified Modeling Language. Class diagrams and the use case diagrams were particularly useful for communication among the development team members. The C++ Standard Template Library provided many useful constructs that would have otherwise involved considerable amount of programming on our part. In order to exploit the multi-thread and socket programming features while maintaining object oriented design we selected the ACE wrapper libraries and TAO CORBA implementation [7]. We have customized TAO CORBA initialization so that it runs with appropriate multi-thread policy, a single name server, etc.. This customization package also provided simpler interface to underlying CORBA calls and provided CORBA name space maintenance. We have also put together a package to provide large file support with improved performance by implementing custom C++ iostream streambuf class.

Logging Manager

The logging manager application is a multi-threaded program implemented with one thread that provides CORBA interface for monitor and control, and a separate thread for each input and output client. This program receives events through input objects, maintains a queue of current and pending events and delivers events out through output objects. Three types of input and output classes are available: socket, CORBA and file. The design is such that any number and type of input or output objects can be connected at any time. The setup of the program is accomplished through CORBA calls. The TCP/IP socket or the file setup information is exchanged through this mechanism. The mutex-locked queue status and the input and output status lists are available for monitoring. The normal mode of operation is of two types, event logging and event distribution, which use intermediate disk files to transfer events.

Event Logging

For event logging, several online event processing daemons will connect to the logging manager and send events through the socket interface. These events are written to the disk files. There is only a small amount of processing involved to handle a selection of special input “events”, begin and end run. The file opening and closing at appropriate time intervals is controlled through these events in the data stream itself. We have typically used 32 Sun Ultra-5 machines on switched 100Base-T network to perform level-3 algorithm and write data to logging manager running on a Sun 4 CPU Enterprise 450 with a large RAID disk and gigabit ethernet interface to the switch. The file output buffers are setup to use large 1 MB read/write buffers achieving about 35 MB/s throughput to the disk. We use 32 kB TCP socket send/receive buffers to optimize network performance. The aggregate throughput to logging manager scales up to about 1 kHz of 35 kB events, i.e., 35 MB/s, with a factor of 10 safety margin. CPU usage at normal 100 Hz operation is less than a 10 % of the machine capacity.

Event Distribution

For prompt reconstruction the events are distributed from a selected disk file to several prompt reconstruction daemons via socket interface. On this socket link there is bidirectional communication to ensure that every event is processed and safely logged to the object database. The objects needed for the constants calculation are accumulated from event data on several prompt reconstruction computers. Only a small amount of processing is involved to drive the prompt reconstruction framework state machine to allow processing these database accumulations from various daemons and constants calculation. All of this support is handled by special marker events inserted into the data stream by the logging manager. The output nodes can join the processing at any time during the processing as the driving of state transitions is done on per output thread basis. Multiple events can be requested by any output link before they acknowledge that the events are processed. Logging manager stores the status of these events so that any events from crashing nodes can be redistributed to other working nodes. To avoid a pathalogical event from killing all the processes, the particular event that is responsible for killing the processor node is marked as such and not redistributed. CORBA interface allows monitoring of the run and output node status. For the prompt reconstruction case, the logging manager has one input file object and up to 200 socket output objects each running in a separate thread. We have not noticed any difficulty in getting sufficient resources to the input file thread when all 202 threads are running. Test runs have yielded throughput of about 400 Hz of 35 kB events, i.e., 14 MB/s including the event acknowledgement handshake. For the production running we allow large granularity, about 3 minutes, database commits, requiring that several hundred megabyte event buffer. In this case, the logging manager was able to supply events at the rate of 125 Hz without throttling input to any nodes.

Conclusions

The commodity Fast Ethernet technology suffices for a majority of our online links and is a good choice considering cost issues. The choice of Gigabit Ethernet for our most demanding links was also appropriate. The object oriented paradigm that we adhered to in building our software enabled its implementation in a timely fashion with good reliability. The ACE and TAO packages make an excellent freeware choice in building object oriented applications in a client-server environment. We also found that the C++ Standard Template Library to be valuable. The performance of this system is scalable well beyond current BABAR requirements.

References

1 The BaBar Technical Design Report, SLAC-R-95-457, 1995 http://www.slac.stanford.edu/BFROOT/doc/TDR.

2 The BaBar Data Acquisition System, I. Scott et al., CHEP 1998.

3 The BaBar Online Computing System, G. Dubois-Felsmann, Paper No. 374, CHEP 2000.

4 Architecture of the BaBar Level-3 Software Trigger, E. D. Frank, CHEP 1998.

5 BaBar Prompt Reconstruction, T. Glanzman, Paper No. 288, CHEP 2000.

6 Network Performance Testing for the BaBar Event Builder, T. J. Pavel et al., CHEP 1998.

7 An Architectural Overview of the ACE Framework, D. C. Schmidt, http://www.cs.wustl.edu/˜schmidt, USENIX login magazine, November, 1998. and The Design of the TAO Real-Time Object Request Broker, D. C. Schmidt et al., Computer Communications, Elsevier Science, Volume 21, No. 4, April, 1998.