The BaBar Online Databases

G. Zioulas[1]*, Yu. Kolomensky*[2]*, S. Metzler*[2]*, V. Miftakov*[3]
for the BaBar Computing Group

[1] Stanford Linear Accelerator Center, Stanford, California 94309, USA
[2] California Institute of Technology, Pasadena, California 91125, USA
[3] Princeton University, Princeton, New Jersey 08544, USA

Abstract

The online databases are presented with emphasis on the design and implementation of the Ambient and Configuration databases. Their performance during the first run of the experiment is discussed. Online database servers and browsers are also described.

Keywords: Database, Online, BaBar, C++, Object Oriented Design

1 Introduction

The BaBar experiment at the Stanford Linear Accelerator Center is designed to study CP violation in decays of B mesons produced in electron-positron interactions. The experiment started running in June 1999. During the first 6 months of operation it has recorded over 1.7 fb^{-1} of integrated luminosity.

BaBar has adopted an object-oriented approach for its online and offline software development with C++ as the principal programming language. To store the acquired data an Object Oriented Database Management System was chosen with a commercial product, Objectivity/DB [1], as the underlying storage technology.

2 The Online Databases

The online system also uses Objectivity/DB for storing time history of the data taking conditions, fast monitoring histograms, ambient data, and configuration parameters of hardware and software. The online databases satisfy the following design requirements:

- Support of object-oriented design and implementation with C++ programming interface;
- Hierarchical structure provides logical storage through persistent objects, databases and federations;
- Application-side caching of objects retrieved from the database improves performance;
- Autonomous partitioning of the federation and fault tolerance makes the data available during network failures;
- Distributed architecture and scalability provides efficient access to Terabytes of data;
- Reuse of existing code developed by the offline system;
- Common tools for administration, maintenance and data distribution.

The online databases consist of three domains: the *Conditions*, the *Ambient* and the *Configuration*.

2.1 Conditions Database

The Conditions database manages and tracks the conditions under which experimental data were acquired. It is primarily used by the offline system to store the detector calibrations, alignment

etc. but it is also used by the online to store the electronic calibrations and the run information. In addition, histograms from the Fast Monitoring are also stored in this database.

A detailed description of the design and implementation is given in reference [2]. Here we describe the Fast Monitoring application.

2.1.1 Fast Monitoring

The BaBar Fast Monitoring system [3] performs automatic comparisons of diagnostic histograms to references. The system runs on 32 Unix computer nodes. The diagnostic data are saved at the completion of each run.

The Fast Monitoring system stores its data in the Fast Monitoring database within the conditions domain. The data are stored as a hierarchical structure that mimics the online data organization. The time interval of the database record is the time interval of the run. Data may be read using a combination of GUI and server tools available through the Distributed Histogramming Package [4] and the Objectivity Browser [5].

The maximum data quantity stored per run is restricted to the shared memory segment limit since data are accessed only through shared memory. That limit is 32 MB. We have approximately 40 runs per day in factory mode, which gives us a maximum storage rate of about 1 GB/day.

2.2 Ambient Database

The Ambient Database is the storage of ambient data which are reported by about 30,000 channels of the Detector Controls system [6]. The channels are reporting asynchronously at rates varying from once every few seconds to several times per second. To reduce the number of database transactions the data is accumulated for a period of one hour. The channels are grouped in objects based on the source of their origin.

The Ambient database is designed so that each detector subsystem will have an index database and a number of object databases. The begin and end times, which define the time interval during which an ambient object is valid, are encapsulated in an interval object along with a pointer to the corresponding ambient object. The interval objects are stored in the index database while the ambient objects are stored in the object databases. The purpose of this separation is to keep the interval database small in size so it can be efficiently searched. The interval objects are indexed with Objectivity's index facility, which allows for rapid lookup of objects. The list of measurements within each ambient object is implemented with variable arrays (ooVArray). Each element of an array contains the value, time and status of the measurement. Another list contains the channel information and pointers to the measurements array. The data for a specific channel is accessed by the time interval, object name and channel number.

There are 27 archiving processes writing about 100 objects in the Ambient database. The archivers flush their data simultaneously once an hour. The total size of the objects is about 2MB. It takes about one minute to store them in the database.

2.2.1 Ambient Server and Browser

The ambient data are retrieved from the database via a Unix server process. The server uses CORBA to distribute the data over the network to Java clients for browsing and analysis. The latest ambient objects entered in the database are translated into CORBA structs and are added to a hierarchical structure, the *Distributed Object Tree*. This tree represents the Ambient database structure. Users can navigate the distributed tree, select channels and display their time history on strip charts. The browser sends a request to the server with the corresponding object name and time interval. The server uses this information to query the database and concatenate the fetched

objects based on the requested time range. The current browser will soon migrate to Java Analysis Studio (JAS) [7], a desktop data analysis application with a rich graphical interface. JAS will allow users to view correlations between different channels and between the ambient and physics data. In the meantime, a C++ application is used to retrieve data from the Ambient database and fill NTUPLES which can then be displayed with PAW.

2.3 Configuration Database

The Configuration database provides storage of the configuring parameters for various hardware and software components of the online system.

The configuration objects are clustered together in containers identified by the subsystem name, class of the object, and an optional secondary key. Each new object is assigned a unique integer index, which allows for efficient retrieval and identification. All objects contain information about the time of creation, creator and a short description. A more detailed description of the API is given in reference [8].

The Configuration database is organized in a tree-like structure that represents the DAQ hierarchy. This tree provides navigation from the top of the hierarchy, indexed by a configuration key, to the subsystem configuration data on the leaves of the tree. The intermediate nodes of the tree are implemented as persistent hash tables where named elements are used to look up the location of the next object along the branch. This organization has the following advantages:

- Ii improves efficiency and performance. Each subsystem only loads configuration objects it needs instead of the whole database;
- It allows authorization control based on subsystem level. Users of various subsystems can work independently without interfering with each other;
- It simplifies partitioning of the system. Different configuration keys can be sent by the Run Control to different partitions.

Proxies are used to retrieve the data from the Configuration database. The proxies perform the database operations and cache configuration objects for repeated access, reducing the database reads and network traffic. A typical *configure* transition takes about 15 sec which includes the distribution of the configuration key to various processes and the retrieval of the objects from the configuration database. Subsequent *configure* transitions of the same run type are significantly faster (a few seconds).

2.3.1 Configuration browser and editor

GUI applications written in C++ were provided to edit and browse the Configuration Trees. The editor allows authorized users to create new or remove old branches of the Configuration Tree corresponding to their subsystems. In addition they can edit a persistent alias map which associates the top-level configuration keys with run types.

2.3.2 Dataflow server

The configuration parameters are made available to the Dataflow [9] Readout Modules (ROMs), which run the VxWorks Tornado operating system, by a dedicated Unix server. The database objects are transported over the network into ROMs in the form of *Tagged Containers* (TC), the self-describing contiguous blocks of data. During a *configure* transition the client code on a ROM receives a configuration key from the Run Control and queries the server with the combination of this key, the TC type and a string secondary key. The server calls the appropriate proxy which fetches the configuration object from the database, converts it to a tagged container and returns it to the ROM.

3 Operational experience and performance

In the BaBar computing architecture, data are acquired by the online system installed at the interaction hall (IR2) and written to files on local disk. The files are transferred over Gigabit ethernet to a farm of SUN Ultra-5 machines at the SLAC Computing Services building where they are processed by the Online Prompt Reconstruction (OPR) and logged to database. To support data taking and event reconstruction a federation was partitioned between the online databases at IR2 and the Event Store at SCS. This allows the DAQ to run even if there are network or system problems at SCS.

Early in the run we found concurrency problems between the online processes trying to write to the Conditions database and the hundreds of OPR processes trying to read back. The interference between OPR and online introduced an unacceptable dead time in our DAQ system. This forced us to split the federation between online and OPR and to periodically sweep the data between the two federations. Modifications to the design of the Conditions database were necessary to allow for online and OPR calibrations to be written in the same database [2]. The Ambient server can also set locks, while processing requests from the browsers, preventing the archivers from flushing their data in the database. However, loss of data is minimized because the server is caching the objects, shortening the legth of its transactions, and the archivers continue to accumulate data if they find the database locked when they try to flush.

After splitting the federations the online database system has been more robust. Cron jobs monitor the state of the online federation and remove locks left by dead processes minimizing down time.

Work is under way to improve the user interfaces and provide better tools for browsing. Additional work is needed to minimize the concurrency problems between the online and OPR and to re-merge the two federations.

4 Acknowledgements

We wish to thank the BaBar database group for their support in implementing and debugging this system. This research is supported by the U.S. Department of Energy.

References

1 Objectivity Inc., Mountain View, CA 94041, USA.
2 I. Gaponenko, “An Overview of the BaBar Conditions Database”, CHEP2000, Padova, Winter 2000.
3 E. Chen *et. al*, “Automated Data Quality Monitoring in the BaBar Online and Offline Systems”, CHEP2000, Padova, Winter 2000.
4 S. Metzler *et. al*, “Distributed Histogramming”, CHEP’98, Chicago, Autumn 1998.
5 A. Adesanya, “An interactive browser for BaBar databases”, CHEP2000, Padova, Winter 2000.
6 J. Olsen, “BaBar Online Detector Control”, CHEP2000, Padova, Winter 2000.
7 A. Johnson *et. al*, “Java Analysis Studio”, CHEP’98, Chicago, Autumn 1998.
8 Yu. Kolomensky *et. al*, “Configuration Database for BaBar Online System” , CHEP’98, Chicago, Autumn 1998.
9 R. Hamilton *et. al*, “The BaBar Data Acquisition System”, CHEP’98, Chicago, Autumn 1998.