SLAC-PUB-9176
September 2001

Optimizing Parallel Access to the BaBar Database System Using CORBA Servers

Jacek Becla[1], Igor Gaponenko[2]

[1]Stanford Linear Accelerator Center
Stanford University, Stanford, CA 94309

[2]Lawrence Berkeley National Laboratory
Berkeley, CA 94720

Presented at CHEP'01: Computing on High-Energy Physics and Nuclear,
9/3/2001—9/7/2001, Beijing, China

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

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

Optimizing Parallel Access to the BaBar Database System Using CORBA Servers

Jacek Becla[1], Igor Gaponenko[2]

[1] Stanford Linear Accelerator Center
[2] Lawrence Berkeley National Laboratory

Abstract

The BaBar Experiment collected around 20 TB of data during its first 6 months of running. Now, after 18 months, data size exceeds 300 TB, and according to prognosis, it is a small fraction of the size of data coming in the next few months. In order to keep up with the data, significant effort was put into tuning the database system. It led to great performance improvements, as well as to inevitable system expansion - 450 simultaneous processing nodes alone used for data reconstruction. It is believed, that further growth beyond 600 nodes will happen soon.

In such an environment, many complex operations are executed simultaneously on hundreds of machines, putting a huge load on data servers and increasing network traffic. Introducing two CORBA servers halved startup time, and dramatically offloaded database servers: data servers as well as lock servers.

The paper describes details of design and implementation of two servers recently introduced in the BaBar system: Conditions OID Server and Clustering Server. The first experience of using these servers is discussed. A discussion on a Collection Server for data analysis, currently being designed is included.

1. Introduction

Few places in the world, if any, can be proud of having a database system as large as the one managed by the BaBar experiment. Up until now, over 300 TB of data were produced and stored in more than 130,000 database files. The system is based on a commercial product – Objectivity/DB [3]. The size of the data is not the only factor that makes the system extremely complex, it also has unusually demanding performance & scalability requirements.

The system has to keep up with data incoming from the detector. In order to achieve that, several hundred computing nodes work 24 hours per day, 7 days a week reconstructing the data and storing it in a persistent format. As of today the Online Prompt Reconstruction (OPR) farm consists of 220 nodes, and it is able to process over 150 pb^{-1} of data (around 5 million of events) per day. It is expected that the number of processing nodes will grow up beyond 600 nodes in the next few months, while the processing rate will double. System growth faster than Moore's Law is expected to continue throughout the lifetime of the experiment – the next 8 years.

Work in a prompt reconstruction system is scheduled in units of *runs*. A run is an arbitrary number of events, typically around 800,000 events. Each node processes a subset of the run. Processing a single run can be split into several phases: startup time, steady processing and close down time. In order to maximize throughput, not only does the steady processing rate has to be maximized, but also startup and close down times should be kept to a minimum. In practice, especially from the database point of view, every node is doing the same work:

1. Startup time:
   - Locating some time intervals specific to the run (stored in Conditions Database [2]) using indexes.
   - Reading the interval containers from the Conditions Database.
   - Reading some metadata that is used to decide where and how to place data (host, file system, database file, container, the database size, the optimum container size, etc).
   - Preparing the placement for the data (make sure all the databases and containers exist, or, if not, create them – usually there are 168 containers in 168 different database files involved).

2. Steady processing:
   - Storing the data persistently in just prepared databases and containers. Occasionally, when a currently used container gets full, finding (or creating) a new one.
3. Close down:
   - Storing some condition information in the Conditions Database.
   - Updating clustering metadata, so that the next run will be able to continue using existing containers and databases.
   - Rolling calibration (spatial and temporal databases).

Most of these operations involve reading and updating the same pieces of the Conditions Database and clustering metadata. In practice, each node usually ends up creating 168 new containers, each in a different database, at the beginning of each run. These multiple requests come almost simultaneously from hundreds of nodes and flood the network and database servers, significantly degrading the efficiency of the system.

In order to alleviate the problems described above, two central CORBA [4] servers have been introduced into the OPR system.

2. Clustering Hint Server

The Clustering Hint Server (CHS) takes care of data placement and clustering. CHS is a multi-threaded, CORBA-based server responsible for managing databases and containers: creation, distributing object ids to clients, and closing full containers in the most optimal way. The server maintains a pool of empty (just pre-created) databases and containers. A client never needs to wait for a creation since there always are available containers. Pre-creation is done in background, if possible, during times when the system is not loaded. During pre-creation, a database and all its containers are pre-sized and disparsified. The former lets a client avoid expensive container extensions as it writes data, the latter de-randomizes disk accesses, greatly improving overall disk throughput of the file systems.

Managing placement centrally carries many additional benefits.

- It greatly reduces network traffic, lock traffic and data server load.
- The server may use much faster algorithms to create large number of containers in a database. In the old model it was impossible to use these algorithms: each client was creating just one container.
- Containers reusing. Thanks to the fact that the server knows which container is in use, it can keep re-assigning containers to clients until they reach optimal size. In the old model keeping track of each container was impossible because access to the metadata became a bottleneck.
- Since containers are reused, they can be pre-sized during creation. In the old model pre-sizing usually meant loosing a lot of disk space. Now, if a container is pre-sized, it does not have to be extended hundreds of times while it is being written. And an extension is a very expensive operation that requires locking the whole database.
- Since the server has a whole picture of the system, it is able to distribute databases across multiple hosts and file systems in the most efficient way – for example, using a round-robin algorithm. Load balancing allows us to maximize the performance of available servers and file systems.
- Since the server knows a state of each container, it can migrate and purge[1] databases immediately after they get full (full databases are never referenced in the OPR system). This feature allows efficiently managed disk space (an expensive resource).
- Pre-creating databases reduces dependency on database creation. Creation requires exclusive access to the database catalog. Should anything go wrong with the creation, the server has a chance to recover before it runs out of databases (in practice it takes many hours before the pool is exhausted).

[1] To migrate is to copy a file to a tape. To purge is to remove a file from a disk.

The server has been in production for several months now. The desired level of server robustness and stability has been achieved within two weeks after launching it. Introducing the server increased processing rate by over 120%.

3. Conditions OID Server

In order to optimize the overall performance of the Conditions Database in the OPR environment we introduced an intermediate caching layer in between the reconstruction jobs and the Objectivity/DB data servers. The wide spectrum of aspects associated with the current design and the implementation of this software layer (OID Server), ranging from its architecture and performance, to the experience gained during the first 6 months of its use in the "production" environment is presented in this section.

The BaBar Conditions Database is an essential component of the overall infrastructure supporting the experiment's data processing and analysis system. This special database provides the detector alignments, calibrations constants, as well as other time-dependent records of the conditions under which the events were taken and processed.

In OPR, the specific organization of the parallel reconstruction code requires the corresponding conditions to be loaded from the Conditions Database by each job before the very first event gets processed. This results in excessive traffic to the Objectivity's data servers and Lock Server at the beginning of the job's execution. We identify this problem as the *startup* one. It can take up to 12 minutes (out of approximately 1 hour of overall processing time for a typical run) for the 150 jobs to begin processing events in a steady rate. Removing this bottleneck has the potential of significantly increasing system performance.

Solution

Our analysis identified the Conditions Database traffic as one of the major contributors to the startup problem. Each jobs loads nearly 200 various conditions, which results in up to 50 MB of network traffic per job. Most of this traffic (nearly 80%) relates to the metadata, including Objectivity's overhead (internal metadata pages for Objectivity's containers and databases) as well as the Conditions Database specific traffic. The ultimate role of operations with the metadata is only to locate the addresses of the conditions objects in the persistent store – object identifiers (OIDs). Luckily, in our design the metadata is physically separated from the condition objects.

Since all jobs do the same work of locating the same (with a chance of over 99%) addresses of condition objects, the solution was to separate the operations with metadata into a dedicated server process called the OID Server and deliver the resulting persistent OIDs of the condition objects to the client jobs. A clear benefit of this approach is that resolved OID (which itself is just 8 byte structure) can be cached in the server's virtual memory.

Implementation

In our implementation we use CORBA as an underlying clients-server communication protocol. When the clients initialize their context, they resolve their server using the CORBA Naming Server. The protocol is flexible enough to allow multiple servers in the same installation. So every time a client communicates with a particular server it may get a command to switch to another server as a part of a server's reply. This makes possible the dynamic load balancing of the servers.

The server itself has a multi-threaded implementation. After experimenting with various multi-threading techniques we decided in favor of the "working thread" approach. The server accumulates its usage statistics and coordinates requests processing between the threads.

A special management and monitoring tool has also been developed. This program talks to the server using the CORBA protocol to change its parameters, control its execution and to acquire its statistics. The advanced GUI version of the monitor written in Java/JFC is also available.

Status and Experience

At the time this paper was being written there were 4 installations of the OID server in production used for the major reprocessing facilities. Some of them have been in use since February 2001. No major robustness or management problems have been seen so far. The total reduction of the overall *startu*p time of the Reconstruction farms is up to 30%. On the other side we can see the dramatic reduction (up to 3 times) of the Conditions Database related traffic between the reconstruction jobs and the corresponding Objectivity/DB servers. Yet the maximum performance of the server far exceeds (by an order of magnitude) the requirements of the current reprocessing facilities. For example, in stress tests the server was able to serve up to 4 kHz of requests while being run with 10 threads on a dedicated 4 CPU machine.

Given the very positive experience with this technique we are also considering applying a similar technique to the data analysis environment. However, this requires developing more advanced transient caching techniques on the server's side since different analysis jobs do not process their events in a coherent manner.

4. Future Plans

Several other places have already been identified, that would clearly benefit from introducing a central management of the metadata.

In the OPR system, should the whole farm crash, recovery is a daunting task. All pending transactions have to be recovered sequentially, and there are no automatic tools that manage that recovery. Currently, each node is trying to recover independently, usually clashing with many others trying to recover at the same time. Central management of the recovery would be the most efficient approach.

Managing event collections is one of the weakest points in the physics analysis system: hundreds of users have to traverse the same tree-node based metadata to locate a single collection. Adding a new collection to the metadata is usually preceded with a long wait for an update lock. A collection server would clearly solve these problems. However, due to lack of manpower, it is expected that a collection server will not be ready for another several months.

5. Summary

Tuning the Online Prompt Reconstruction system is an ongoing process since the BaBar experiment started to take data in May 1999. After the first year, it became progressively more and more difficult to further optimize an already well-tuned system [1]. At the same time the accelerator significantly exceeded its designed goals, making even higher demands on the OPR system. In order to meet the expectations, new ways of speeding up the system had to be found.

Introduction of two new central servers was a relatively non-trivial change preceded by several months of design and implementation. Introducing it without disrupting the whole system and maintaining backwards compatibility made the task even more complex. Now, having achieved these goals, there is no doubt that it was worth doing. The success of the clustering hint server and conditions OID server encouraged us to go even further toward central management of the system. A recovery server is being developed now, and a collection server will be developed in the near future.

Prognosis seems to indicate that the incoming luminosity will continue its growth for at least the next five years. In order to continue keeping up with data, the OPR system has to continuously evolve, and new ways of improving its performance have to be found.

6. References

1. Becla J., *Improving Performance of Object Oriented Databases, BaBar Case Studies*, CHEP'00, Padova, Italy, January 2000
2. Gaponenko I., *An Overview of the BaBar Conditions Database*, CHEP'00, Padova, Italy, January 2000
3. http://www.objectivity.com
4. http://www.omg.com