Scalability to Hundreds of Clients in HEP Object Databases

Koen Holtman

CERN – EP division
CH - 1211 Geneva 23, Switzerland

Julian Bunn

Currently on special leave from CERN at:
256-48 HEP, Caltech
1200 E. California Blvd.
Pasadena, CA 91125, USA

The current offline computing strategy in the CMS experiment at CERN depends crucially on the use of a scalable object database. We have tested the scalability of the Objectivity/DB object database under CMS data acquisition and event reconstruction workloads. We obtained almost ideal scaling results with loads up to 240 active clients.

1 Introduction

The CMS collaboration plans to implement its data storage and processing system using a single large federated object database [1]. The scalability of such a database is an important consideration. In this respect, the main goals for the CMS system are

- Total storage of several Petabytes
- Aggregate throughput at least a few times higher than the DAQ (data acquisition) rate of 100 MB/s
- Hundreds of independent database clients.

These scalability issues are being studied by CMS as part of the RD45 collaboration [2] and in the context of the GIOD project [3], which is a joint project between Caltech, HP, and CERN. In this paper, we report on scalability tests of the Objectivity/DB object database [4] made on the 256-processor HP Exemplar machine at Caltech. Our tests focused on the behaviour of the throughput as a function of the number of database clients, under DAQ and reconstruction style workloads.

2 Testing platform and software

The scalability tests were performed on the HP Exemplar machine at Caltech, a 256 CPU SMP machine of some 0.1 TIPS. The machine consists of 16 nodes, which are connected by a special-purpose fast network called a CTI (see figure 1). Each node contains 16 PA8000 processors and one node file system. A node file system consists of 4 disks with 4-way striping, with a file system block size

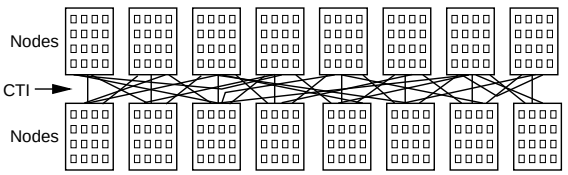

Figure 1: Configuration of the HP Exemplar at Caltech

of 64 KB and a maximum raw I/O rate of 22 MB/s. We used up to 240 processors and up to 8 node file systems in our tests. Our datasets always came from disk, never from the file system cache. An analysis of the raw I/O behaviour of the Exemplar can be found in [5].

The Exemplar runs a single operating system image, and all node file systems are visible as local UNIX file systems to any process running on any node. If the process and file system are on different nodes, data is transported over the CTI. The CTI was never a bottleneck in the test loads we put on the machine: it was designed to support shared memory programming and can easily achieve data rates in the GB/s range. As such, the Exemplar can be thought of as a farm of 16 16-processor UNIX machines with cross-mounted file systems, and an infinite capacity network. Though the Exemplar is not a good model for current UNIX or PC farms, where network capacity is a major constraining factor, it could in fact be a good model for future farms which use GB/s networks like Myrinet [6] as an interconnect.

The object database tested was the HP-UX version of Objectivity/DB v4.0.10 [4]. Our test setup did not use the so-called Objectivity AMS server for remote database access: all database file I/O was directly between the database clients and the operating system. In [7], it is reported that major scalability problems can be expected if the current AMS version is used. We did not invoke any Objectivity FTO/DRO features. The test loads were generated with the TOPS framework [8] which runs on top of Objectivity.

Two things in the Objectivity architecture were of particular concern. First, Objectivity does not support a database page size of 64 KB, it only supports sizes up to 64 KB minus a few bytes. Thus, it does not match well to the node file systems which have a block size of exactly 64 KB. After some experiments we found that a database page size of 32 KB was the best compromise, so we used that throughout our tests. Second, the Objectivity architecture uses a single lock server process to handle all locking operations. This lock server could become a bottleneck when the number of (lock requests from) clients increases. The Objectivity DRO/FTO option does allow one to run multiple lock servers, but also adds communication between lock servers. We currently do not know whether FTO/DRO can be exploited to avoid lock server hot spots.

3 Reconstruction test

We have tested the database under an event reconstruction[1] workload with up to 240 clients. In this workload, each client runs a simulated reconstruction job on its own set of events. For one event, the actions are as follows:

- Reading: 1 MB of 'raw' data is read, as 100 objects of 10 KB. The objects are read from 3 containers: 50 from the first, 25 from the second, and 25 from the third. Inside the containers, the objects are clustered sequentially in the reading order.
- Writing: 100 KB of 'reconstructed' data is written, as 10 objects of 10KB, to one container.
- Computation: $2 * 10^3$ MIPSs are spent per event, this corresponds to 5 CPU seconds on one Exemplar CPU.

Reading, writing, and computing are all done interleaved with each other. The data sizes are derived from the CMS computing technical proposal [1]. The proposal predicts a computation

[1] In high energy physics, an event is an abstraction which corresponds to the occurrence of collisions between particles inside a physics detector. Event reconstruction is the process of computing physical interpretations (reconstructed data) of the raw event data measured by the detector.

time of $2 * 10^4$ MIPSs per event. However, it also predicts that CPUs will be 100 times more powerful (in MIPS per $) at CMS startup in 2005. We expect that disks will only be a factor 4 more powerful (in MB/s per $) in 2005. In our test we chose a computation time of $2 * 10^3$ MIPSs per event as a compromise. The clustering strategy for the raw data is based on [9]. The detector is divided into three separate parts and data from different parts are clustered separately in different containers. This allows for faster access in analysis efforts which need some parts of the detector only. Note that this arrangement is similar to the one used for reconstructed data in BaBar [10] [11]. The database files are divided over four node file systems, with the federation catalog and the journal files on a fifth file system. In reading the raw data, we used the read-ahead optimisation described in section 4.

The test results are shown in figure 2. The solid curve shows the aggregate throughput for the CMS reconstruction workload described above. The aggregate throughput (and thus the number of events reconstructed per second) scales almost linearly with the number of clients. In the left part of the curve, 91% of the allocated CPU resources are spent running actual reconstruction code. With 240 clients, 83% of the allocated CPU power (240 CPUs) is used for physics code, yielding an aggregate throughput of 47 MB/s (42 events/s), using about 0.1 TIPS.

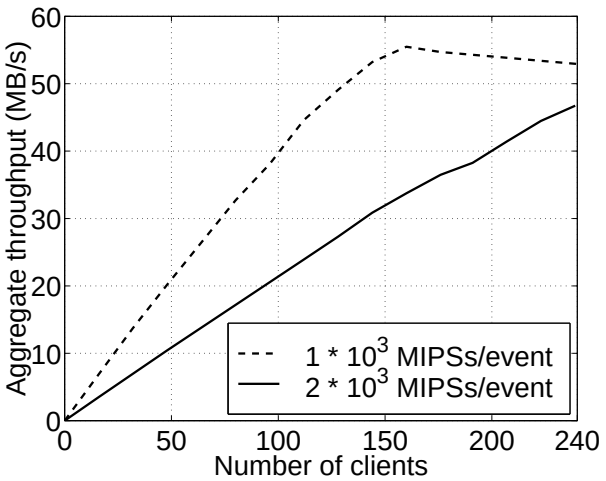

Figure 2: Scalability of reconstruction workloads

The dashed curve in figure 2 shows a workload with the same I/O profile as described above, but half as much computation. This curve shows a clear shift a from CPU-bound to a disk-bound workload at 160 clients. The maximum throughput is 55 MB/s, which is 63% of the maximum raw throughput of the four allocated node file systems (88 MB/s). Overall, the disk efficiency is less good than the CPU efficiency. The mismatch between database and file system page sizes discussed in section 1 is one obvious contributing factor to this. In tests with fewer clients on a platform with a 16 KB file system page size, we have seen higher disk efficiencies for similar workloads.

4 The read-ahead optimisation

When reading raw data from the containers in the above reconstruction tests, we used a read-ahead optimisation layer built into our testbed. The layer takes the form of a specialised iterator, which causes the database to read containers in bursts of 4 MB (128 pages) at a time. Without this layer, the (simulated) physics application would produce single page reads interspersed with computation. Tests have shown that such less bursty reading leads to a loss of I/O performance.

In [9] we discussed I/O performance tests for a single client iterating through many containers, with and without the read-ahead optimisation. Here, we will consider the case of N clients all iterating through N containers, with each client accessing one container only. The computation in each client is again $2 * 10^3$ MIPSs per MB read. Containers are placed in databases on

two node file systems, which have a combined raw throughput of 44 MB/s.

Figure 3 shows that without the read-ahead optimisation, the workload becomes disk-bound fairly quickly, at 64 clients. Apparently, a lot of time is lost in disk seeks between the different containers. In this test, the lack of a read-ahead optimisation degrades the maximum I/O performance with a factor of two. Because of the results in [9], we expect that the performance would have been degraded even more in the reconstruction test of section 3, where each client reads from three containers.

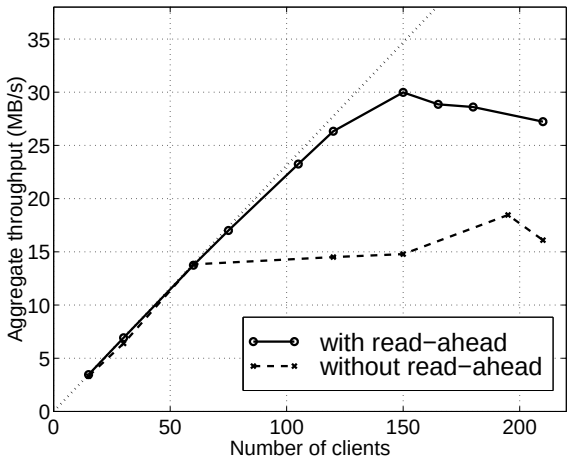

Figure 3: Performance of many clients all performing sequential reading on a container

5 DAQ test

We do not currently advocate the use of an object database as the primary storage method in a real-time DAQ system. We feel that currently, the most attractive approach still is to stream data to flat files, and to then convert these files into objects in quasi-realtime. We have tested the database with such a quasi-realtime data acquisition workload up to 238 clients.

In this test, each client is writing a stream of 10 KB objects to its own container. For every event (1 MB raw data) written, about 180 MIPSs (0.45 CPU seconds on the Exemplar) are spent in simulated data formatting. For comparison, 0.20 CPU seconds are spent by Objectivity in object creation and writing, and the operating system spends 0.01 CPU seconds per event. No read operations on flat files or network reads are done by the clients. The database files are divided over eight node file systems, with the federation catalog and the journal files on a ninth file system.

The test results are shown in figure 4. Again we see a transition from a CPU-bound to a disk-bound workload. The highest throughput is 145 MB/s at 144 clients, which is 82% of the maximum raw throughput of the eight allocated node file systems (176 MB/s).

In workloads above 100 clients, when the node file systems become saturated with write requests, these file systems show some surprising behaviour. It can take very long, up to minutes, to do basic operations like syncing a file (which is done by the database when committing a transaction) or creating a new (database) file. We believe this is due

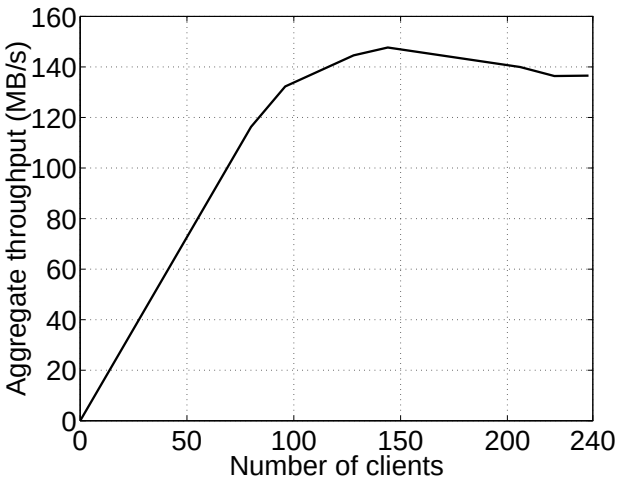

Figure 4: Scalability of a DAQ workload

to the appearance of long 'file system write request' queues in the operating system. During the test, other file systems not saturated with write requests still behave as usual. We conclude from this that one should be careful in saturating file systems with write requests: unexpected long slowdowns may occur.

6 Client startup

We measured the scalability of client startup times throughout our tests. We found that the client startup time depends on the number of clients already running and on the number of clients being started at the same time. It depends much less on the database workload, at least if the federation catalog and journal files are placed on a file system that is not heavily loaded. With heavily loaded catalog and journal file systems, startup times of many minutes have been observed.

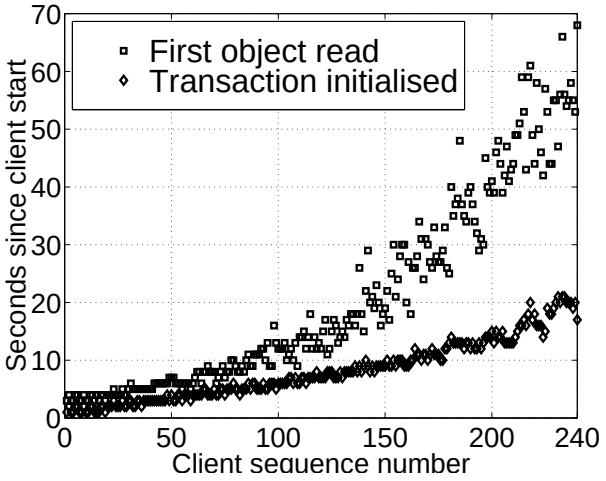

Figure 5: Client startup in the $1 * 10^3$ MIPSs reconstruction test

Figure 5 shows a startup time profile typical for our test workloads. Here, new clients are started in batches of 16. For client number 240, the time needed to open the database and initialise the first database transaction is about 20 seconds. The client then opens four containers (located in three different database files), reads some indexing data structures, and initialises its reconstruction loop. Some 60 seconds after startup, the first raw data object is read. If a single new client number 241 is started by itself, opening the database and initialising the transaction takes some 5 seconds.

Overall, we feel that the startup times scale reasonably well. However, the curve does show that it would be attractive, especially for interactive systems with many users, to keep the database clients running all the time, and to send commands to them as a way of starting new jobs. Another advantage of this approach would be that small frequently used datasets could be kept permanently in client cache memory.

7 The lock server

In all tests described above, we found that the Objectivity lock server was not a bottleneck. Writing will load the lock server more than reading. An Objectivity client will contact the lock server whenever it resizes a container which is being filled. In our tests we used a large initial container size (200 pages) and container growth factor (20%). With smaller growth factors, the lock server will be contacted more often, and could become a bottleneck. From a study of lock server behaviour under artificial database workloads with a high rate of locking, we estimate that lock server communication may become a bottleneck in a DAQ scenario above 1000 MB/s.

Note that we have not studied interactive analysis workloads with hundreds of physicists using the same federated database. We expect that such workloads will be much more challenging as far as lock server communication is concerned. Simultaneous use by hundreds of users also

introduces read and update operations on shared data, for example on a global name space. We have not tested the scalability of updating shared data. Finally, it should be noted that in our tests, the lock server was either on the same machine as the clients, or nearby on a fast LAN. Little is known about the performance effects of using a far-away lock server.

8 Conclusions

Objectivity shows almost ideal scalability, up to 240 clients, under CMS reconstruction and DAQ workloads. We found excellent utilisation of allocated CPU resources, and reasonable to good utilisation of allocated disk resources on our test platform. It should be noted that our test platform has a very fast internal network. On cross-mounted PC or workstation farms using current network technology, network related scalability problems may appear. Major scalability problems can be expected if the current AMS version is used.

Our reconstruction tests validate current ideas about object clustering in a farming configuration [9] [10] [11]. A read-ahead optimisation is needed to get reasonable disk efficiency. Taking expected hardware developments into account, our work provides a proof-of-concept implementation, which shows that it will be possible to run all CMS full reconstruction jobs against a single ODMG federation containing all raw and reconstructed data.

References

[1] CMS Computing Technical Proposal. CERN/LHCC 96-45, CMS collaboration, 19 December 1996.

[2] RD45, A Persistent Storage Manager for HEP. http://wwwcn.cern.ch/asd/cernlib/rd45/

[3] The GIOD project, Globally interconnected object databases. http://pcbunn.cithep.caltech.edu/

[4] Objectivity/DB. Vendor homepage: http://www.objy.com/

[5] R. Bordawekar, Quantitative Characterization and Analysis of the I/O behavior of a Commercial Distributed-shared-memory Machine. CACR Technical Report 157, March 1998. To appear in the Seventh Workshop on Scalable Shared Memory Multiprocessors, June 1998. See also http://www.cacr.caltech.edu/~rajesh/exemplar1.html

[6] Myrinet network products. Vendor homepage: http://www.myri.com/

[7] A. Hanushevsky, Developing Scalable High Performance Terabyte Distributed Databases. Proc. of CHEP'98, Chicago, USA.

[8] TOPS, Testbed for Objectivity Performance and Scalability, V1.0. Available from http://wwwcn.cern.ch/~kholtman/

[9] K. Holtman, Clustering and Reclustering HEP Data in Object Databases. Proc. of CHEP'98, Chicago, USA.

[10] D. R. Quarrie et al, First Experience with the BaBar Event Store Proc. of CHEP'98, Chicago, USA.

[11] J. Becla, Data clustering and placement for the BaBar database. Proc. of CHEP'98, Chicago, USA.