Object Database Scalability for Scientific Workloads

Technical Report

Julian J. Bunn[1] Koen Holtman[2], Harvey B. Newman[1]

[1] 256-48 HEP, Caltech, 1200 E. California Blvd., Pasadena, CA 91125, USA

[2] CERN EP-Division, CH - 1211, Geneva 23, Switzerland

We describe the PetaByte-scale computing challenges posed by the next generation of particle physics experiments, due to start operation in 2005. The computing models adopted by the experiments call for systems capable of handling sustained data acquisition rates of at least 100 MBytes/second into an Object Database, which will have to handle several PetaBytes of accumulated data per year. The systems will be used to schedule CPU intensive reconstruction and analysis tasks on the highly complex physics Object data which need then be served to clients located at universities and laboratories worldwide. We report on measurements with a prototype system that makes use of a 256 CPU HP Exemplar X Class machine running the Objectivity/DB database. Our results show excellent scalability for up to 240 simultaneous database clients, and aggregate I/O rates exceeding 150 Mbytes/second, indicating the viability of the computing models.

1 Introduction

The Large Hadron Collider (LHC) is currently under construction at the European Laboratory for Particle Physics in CERN, Geneva, Switzerland. Due to start operation in 2005, the LHC will collide particles (protons) at energies up to 14 TeV, the highest energy collisions yet achieved. Analysis of the collisions will hopefully uncover the Higgs particle, which is believed to be responsible for giving all other particles their mass. Finding the Higgs, or proving that it does not exist, is currently the 'Holy Grail' of particle physics.

Collisions in the LHC are expected to occur at a rate of about 800 million per second. Of these millions of events, only about 100 (or 0.0000001 percent) are expected to reveal the Higgs particle. The collisions take place inside massive detectors, whose task is to identify and select these candidate events for recording, a process called 'triggering'. The triggering rate in the two main LHC detectors is expected to be approximately 100 Hz. Each candidate event is comprised of approximately 1 MByte of combined data from the very many sub-elements of the detector. The 'raw' event data thus emerge from the detector's electronic data acquisition (DAQ) system at a rate of around 100 MBytes per second.

The raw event data at the LHC will amount to several PetaBytes (10^{15} bytes) per year, each year for the estimated twenty year lifetime of the experiments. The data are already highly compressed

when they emerge from the DAQ system, and they must be stored in their entirety. From these data, reconstructions of physics 'objects', such as tracks, clusters and jets will take place in near real time on dedicated processor farms of an estimated 10^7 MIPS. The reconstructed objects will add about 200 kBytes of extra information to each event. By the time the LHC programme reaches maturity, projections indicate that the total event data volume will be in excess of 100 PetaBytes.

Managing this quantity of data, and making it available to the large multinational community of physicists participating in the CERN Physics Programme, is an unprecedented computing challenge. It is a tenet of the community that physicists working at institutes remote from CERN should enjoy the same level of access to the data as their colleagues located at CERN. This imposes the condition on the LHC Computing Models that either the data be continuously transported across the network or that analysis tasks (or queries) be moved as close to the data as possible. In practice, rapid decisions on whether to move the data in the network, or move the task, have to be made.

To tackle the scale and complexity of the data, the currently favoured technologies for the LHC Computing Models include Object Oriented software to support the data model, distributed Object Database Management Systems (ODBMS) to manage the persistency of the physics objects, and Hierarchical Storage Management systems to cope with the quantity of data, and support access to 'hot' and 'cold' event data.

The GIOD (Globally Interconnected Object Databases) Project [2], a joint effort between Caltech, CERN and Hewlett Packard Corporation, has been investigating the use of WAN-distributed Object Databases and Mass Storage systems for LHC data. We have been using several key hardware and software technologies for our tests, including a 256 CPU Caltech HP Exemplar of 0.1 TIPS, the High Performance Storage System (HPSS) from IBM, the Objectivity/DB ODBMS, and various high speed Local Area and Wide Area networks. One particular focus of our work has been on measuring the capability of the Object Database to

- support hundreds of simultaneous clients
- allow reading and writing at aggregate data rates of ¿ 100 MBytes/second
- scale to the complexity and size of the LHC data.

In this paper, we report on scalability tests of the Objectivity/DB object database [3] made on the 256-processor HP Exemplar located at Caltech's Center for Advanced Computing Research. Our tests focused on the behaviour of the aggregate throughput as a function of the number of database clients, under various representative workloads, and using realistic (simulated) LHC event data

2 Testing platform and Object database

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 of 64 KB and a maximum raw I/O rate of 22 MBytes/second. We used up to 240 processors and up to 15 node file systems in our tests. We ensured that data was always read from disk, and never from the file system cache. An analysis of the raw I/O behaviour of the Exemplar can be found in [4].

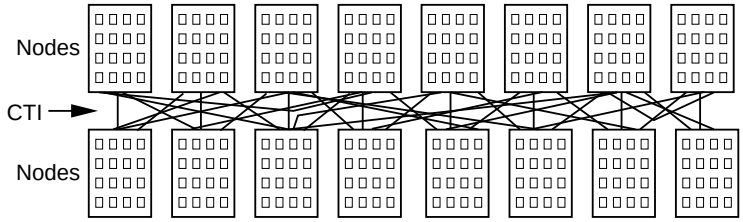

Figure 1: Configuration of the HP Exemplar at Caltech

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 GBytes/second range. As such, the Exemplar can be thought of as a farm of sixteen 16-processor UNIX machines with cross-mounted file systems, and a semi-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 is perhaps a good model for future farms which use GBytes/second networks like Myrinet [5] as an interconnect.

The object database tested was the HP-UX version of Objectivity/DB [3]. The Objectivity/DB architecture comprises a 'federation' of databases. All databases in the federation share a common Object scheme, and are indexed in a master catalog. Each database is a file. Each database contains one or more 'containers'. Each container is structured as a set of 'pages' (of a unique size) onto which the persistent objects are mapped. The database can be accessed by clients which are applications linked against the Objectivity/DB libraries and the database schema files. Client access to local databases is achieved via the local file system. Access to remote databases is made via an 'Advanced Multithreaded Server' which returns database pages across the network to the client. Database locks are managed by a lockserver process. Locks operate at the database container level.

We report on two sets of tests, completed with different database configurations and different data.

3 Tests with synthetic data

Our first round of tests used synthetic event data represented as sets of 10 KByte objects. A 1 MByte event thus became a set of 100 objects of 10 KB. Though not realistic in terms of

physics, this approach does have the advantage of giving cleaner results by eliminating some potential sources of complexity.

For these tests we used Objectivity/DB v4.0.10. We placed all database elements (database clients, database lockserver, federation catalog file) on the Exemplar itself. Database clients communicated with the lockserver via TCP/IP sockets, but all traffic was local inside the supercomputer. The federation catalog and the payload data were accessed by the clients though the Exemplar UNIX filesystem interface.

The test loads were generated with the TOPS framework [6] 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 lockserver process to handle all locking operations. This lockserver could become a bottleneck when the number of (lock requests from) clients increases.

3.1 Reconstruction test

In particle physics, an 'event' occurs when two particles collide inside a physics detector. Event reconstruction is the process of computing physical interpretations (reconstructed data) of the raw event data measured by the detector.

We have tested the database under an event reconstruction 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 (equivalent to 5 CPU seconds on one Exemplar CPU).

Reading, writing, and computing are interleaved with one another. The data sizes are derived from the CMS computing technical proposal [1]. The proposal predicts a computation time of $2*10^4$ MIPSs per event. However, it also predicts that CPUs will be 100 times more powerful (in MIPS per $) at LHC startup in 2005. We expect that disks will only be a factor 4 more powerful (in MBytes/second 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 [7]. The detector is divided into three separate parts and data from each part are clustered separately in different containers. This allows faster access for analysis tasks which only require need some parts of the detector. The database files are divided over four Exemplar 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.

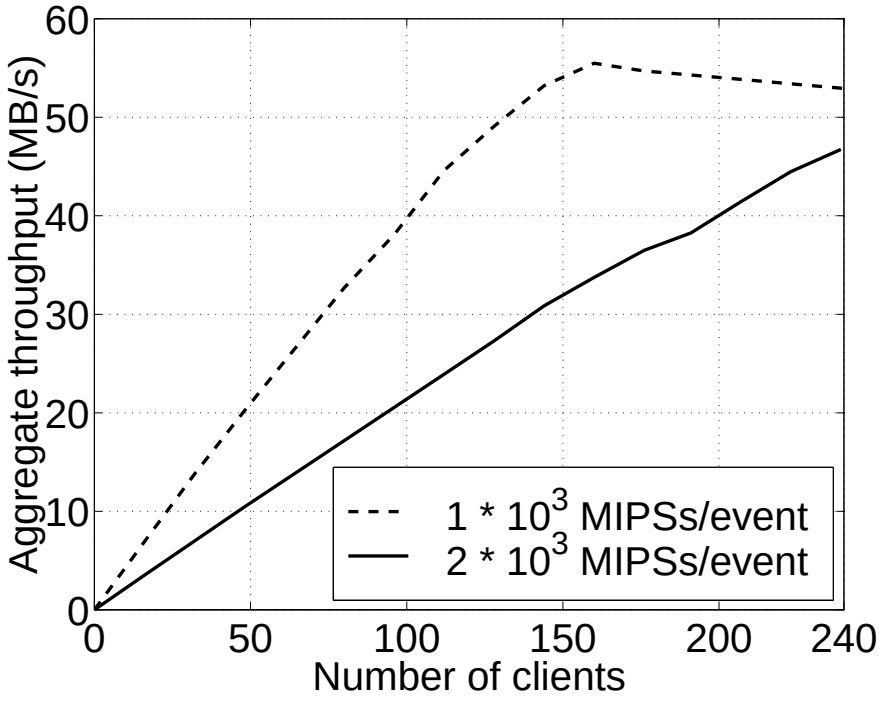

Figure 2: Scalability of reconstruction workloads

The results from our tests 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 MBytes/second (42 events/s), using about 0.1 TIPS.

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 MBytes/second, which is 63% of the maximum raw throughput of the four allocated node file systems (88 MBytes/second). Overall, the disk efficiency is less good than the CPU efficiency. The mismatch between database and file system page sizes discussed in section 2 is one obvious contributing factor to this. In tests with fewer clients on a platform with a 16 KByte file system page size, we have seen higher disk efficiencies for similar workloads.

3.2 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 MByte (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 [7] 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 Megabyte read. Containers are placed in databases on two node file systems, which have a combined raw throughput of 44 MBytes/second.

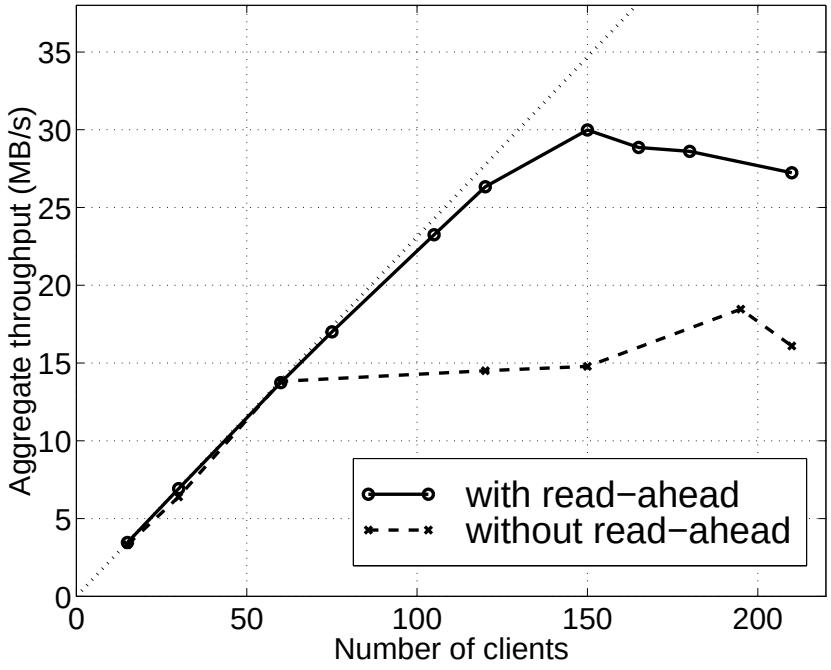

Figure 3: Performance of many clients all performing sequential reading on a container

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 [7], 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.

3.3 DAQ test

In this test, each client is writing a stream of 10 KByte objects to its own container. For every event (1 MByte 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.

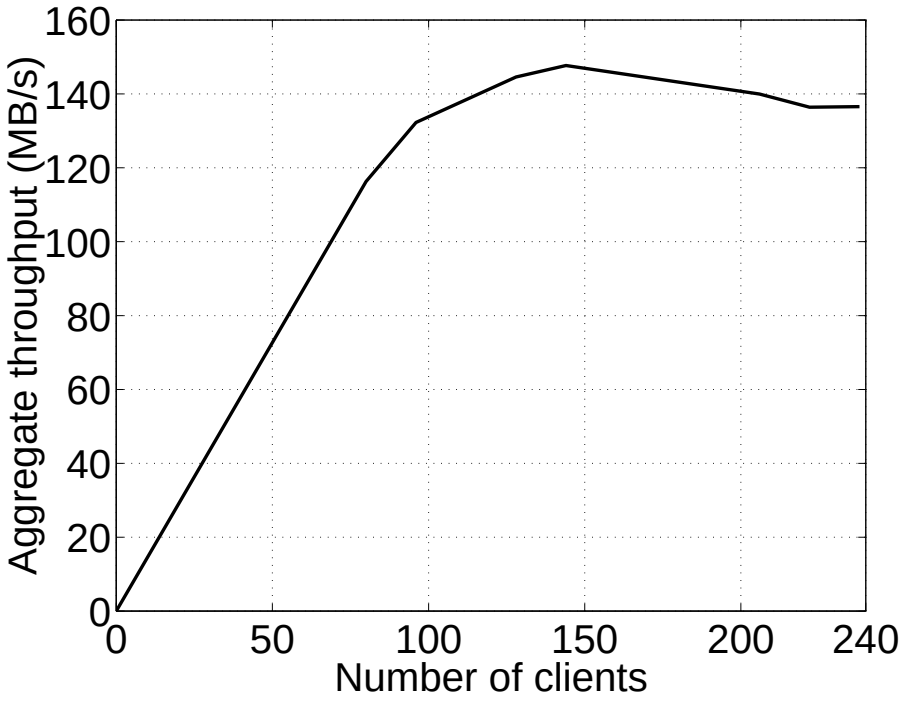

Figure 4: Scalability of a DAQ workload

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 MBytes/second at 144 clients, which is 82% of the maximum raw throughput of the eight allocated node file systems (176 MBytes/second).

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, several minutes, to perform 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 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.

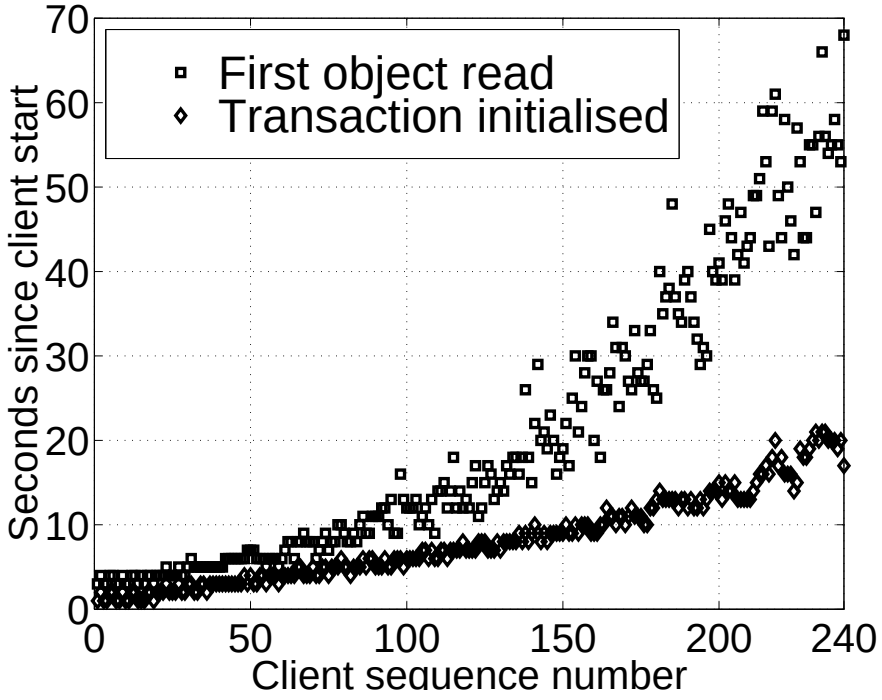

Figure 5: Client startup in the $1 * 10^3$ MIPSs reconstruction test

3.4 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 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.

4 Tests with real physics data

Our second round of tests wrote realistic physics event data into the database. These data were generated from a pool of around one million fully simulated LHC multi-jet QCD events (figure

6). The simulated events were used to populate the Objectivity database according to an object scheme that fully implemented the complex relationships between the components of the events. The average size of the events used in the tests was 260 KB.

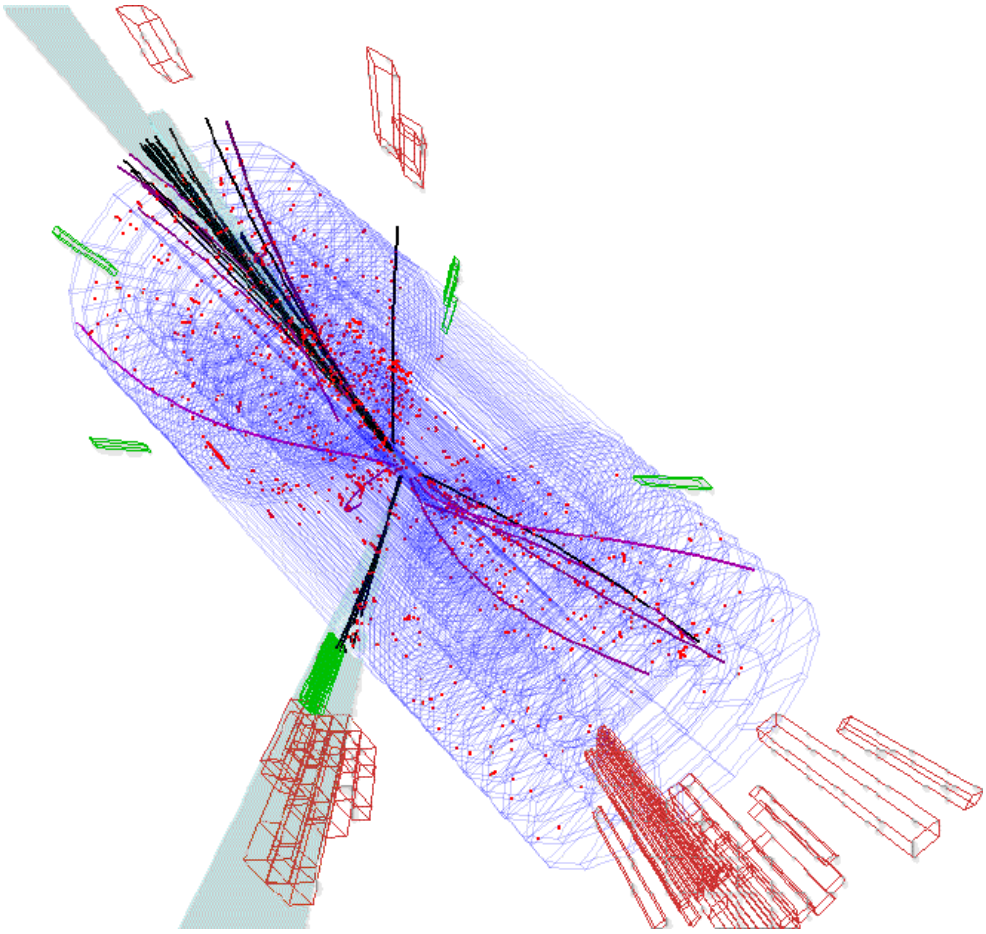

Figure 6: A typical event with its tracks, detector space points and energy clusters

In these tests we used Objectivity/DB v5.0. Only the database clients and the payload database files were located on the Exemplar system. The lockserver was run on an HP workstation connected to the Exemplar via a LAN. The database clients contacted the lockserver over TCP/IP connections. The federation catalog was placed on a C200 HP workstation, connected to the Exemplar over a dedicated ATM link (155 Mbits/second). The clients accessed the catalog over TCP/IP connections to the Objectivity/DB AMS server, which ran on the C200 workstation.

The event data in these tests were written using the software developed in the GIOD project [2]. Each database client first read 12 events into memory, then wrote them out repeatedly into its own dedicated database file. Once the database file reached a size of about 600 MBytes, it was closed and deleted by the client. Then the client created and filled a new database file. This was arranged to avoid exhausting file system space during the tests. In a real DAQ system, periodic switches to new database files would also occur, whilst retaining the old database files.

Database files were evenly distributed over 15 node file systems on the Exemplar. Of these node file systems, ten contain 4 disks are rated at 22 Mbytes/second raw, the remaining five contain fewer disks and achieve a lower throughput. The 15 node filesystems used contain 49 disks in total.

We used two different data models for the event data to be written:

- In one set of tests, we wrote data in the 'GIOD data model' which is the data model developed in the GIOD project [2]. In this data model, a raw event consists of 6 objects, each of these objects placed in a different Objectivity container in the same database file, and with object associations (links) between these objects. The objects in the GIOD data model are shown in figure 7.

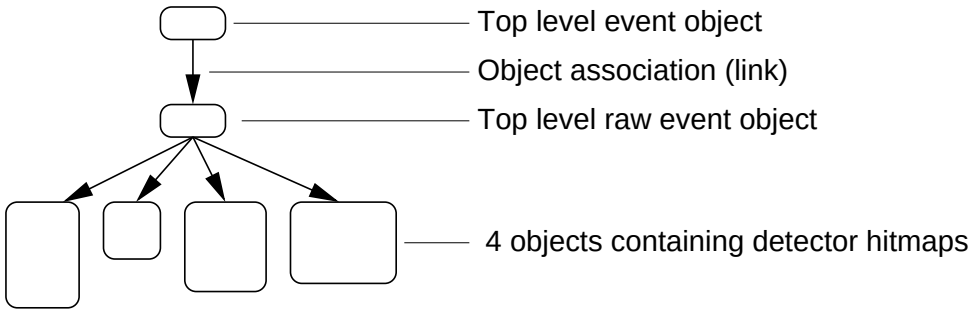

Figure 7: Objects and their relations in the GIOD data model

- We ran another set of tests to quantify the overheads associated with the GIOD event data model. These tests used a simpler '1–container data model', in which all 6 objects in the GIOD raw event were written to a single container, without object associations being created.

4.1 Test results

We ran tests with 15, 30, and 45 database clients writing events concurrently to the federated database, with the two different data models discussed above. Figure 8 shows the test results. The 1–container data model shows a best aggregate throughput rate of 172 MBytes/second, reached with 45 running clients. With the GIOD data model a rate of 154 MBytes/second was achieved when running with 30 clients. We note that the overhead associated with the GIOD model event structure is not significant.

4.2 Analysis of the scaling limit in figure 8

In the earlier tests with synthetic data (section 3), the scaling curves flatten out, when more clients are added, because of limits to the available disk bandwidth on the Exemplar. In the real physics

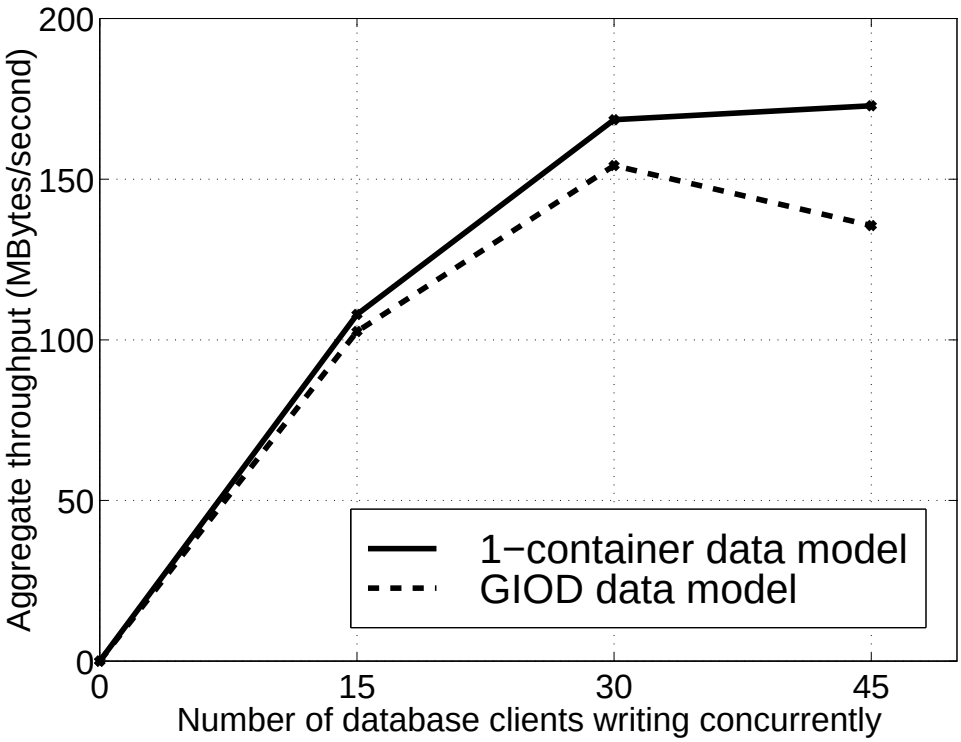

Figure 8: DAQ tests with real physics data

data tests of figure 8, the curves flatten out before the available disk bandwidth is saturated. In this case we found that an access bottleneck to the federation catalog file was the limiting factor.

In the tests of figure 8, the catalog file is located remotely on a C200 workstation connected to the Exemplar with an ATM link. A client needs to access the catalog file whenever it creates a new database, and whenever it deletes a database after closing it on reaching the 600 MB limit discussed above. Throughout our tests, we found that no more than about 18 pairs of 'delete and create new database' actions could be performed every minute. This was irrespective of the number of clients running: in related tests we ran with up to 75 clients, and observed that only about 30 to 45 clients were actively writing to the database at the same time. All remaining clients were busy waiting for their turn to access the remote catalog file.

The bottleneck in access to the remote catalog file was caused by a saturation of the single CPU on the C200 workstation holding the catalog. The AMS server process on the C200, which provided remote access to the catalog file, used only some 10–20% of the available CPU time. The remainder of the CPU time was spent in kernel mode, though we are not sure on what. The dedicated ATM link between the C200 workstation and the Exemplar was not saturated during our tests: peak observed throughputs were 1.2 MBytes/second, well below its 155 Mbits/second capacity. Most (80%) of the ATM traffic was towards the Exemplar system, consistent with the database clients reading many index pages from the catalog file, and updating only a few.

An obvious way to improve on the scaling limit is to create larger database files, or to put the

catalog file locally on the Exemplar system, as was done in the tests with synthetic data (section 3). Another option is to create a large number of empty database files in advance.

5 The lockserver

The lockserver, whether run remotely or locally on the Exemplar, was not a bottleneck in any of our tests. From a study of lockserver behaviour under artificial database workloads with a high rate of locking, we estimate that lockserver communication may become a bottleneck in a DAQ scenario above 1000 MBytes/second.

6 Conclusions

In the first series of tests, with all components of the Objectivity/DB system located on the Exemplar, we observed almost ideal scalability, up to 240 clients, under synthetic physics reconstruction and DAQ workloads. The utilisation of allocated CPU resources on the Exemplar is excellent, with reasonable to good utilisation of allocated disk resources. It should be noted that the Exemplar has a very fast internal network.

In the second series of tests the database clients were located on the Exemplar, and the Objectivity lockserver, AMS and catalog were located remotely. In this configuration, the system achieved aggregate write rates into the database of more than 170 MBytes/second. This exceeds the 100 MBytes/second required by the DAQ systems of the two main LHC experiments.

Our measurements confirm the viability of using commercial Object Database Management Systems for large scale particle physics data storage and analysis.

References

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

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

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

[4] 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

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

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

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