The GIOD Project -
Globally Interconnected Object Databases

Julian J. Bunn[1,2], *Koen Holtman*[2], *Harvey B. Newman*[1], *Richard P. Wilkinson*[1]

[1] California Institute of Technology, Pasadena, USA
[2] CERN, Geneva, Switzerland

Abstract

The GIOD (Globally Interconnected Object Databases) Project, a joint effort between Caltech and CERN, funded by Hewlett Packard Corporation, has investigated the use of WAN-distributed Object Databases and Mass Storage systems for LHC data. A prototype small-scale LHC data analysis center has been constructed using computing resources at Caltechs Centre for Advanced Computing Research (CACR). These resources include a 256 CPU HP Exemplar of ˜ 4600 SPECfp95, a 600 TByte High Performance Storage System (HPSS), and local/wide area links based on OC3 ATM. Using the Exemplar, a large number of fully simulated CMS events were produced, and used to populate an object database with a complete schema for raw, reconstructed and analysis objects. The reconstruction software used for this task was based on early codes developed in preparation for the current CMS reconstruction program, ORCA. Simple analysis software was then developed in Java, and integrated with SLACs Java Analysis Studio tool. An event viewer was constructed with the Java 3D API. Using this suite of software, tests were made in collaboration with researchers at FNAL and SDSC, that focused on distributed access to the database by numerous clients, and measurements of peak bandwidths were made and interpreted. In this paper, some significant findings from the GIOD Project are presented, such as the achievement of the CMS experiment's 100 MB/sec database I/O milestone.

Keywords: GIOD,ODBMS,WAN,LHC,CMS

1 Introduction

The GIOD Project was initiated in late 1996 by Caltech, CERN and Hewlett Packard Corporation [1] The project was to address the key issues of wide area network-distributed data access and analysis for the next generation of high energy physics experiments. During the course of the project, we have benefitted from collaboration with the RD45 Project, FermiLab, San Diego Supercomputer Centre, and colleagues in the MONARC and PPDG projects.

Rapid and sustained progress has been achieved since 1996: we have built prototype databases and reconstruction, analysis and (Java3D) visualization systems. This has allowed us to test, validate and develop some strategies and mechanisms that will make the implementation of massive distributed systems for data access and analysis in support of the LHC physics program possible. These systems will be dimensioned to accommodate the volume (measured in PetaBytes) and complexity of the data, the geographical spread of the institutes and the large numbers of physicists participating in each experiment. At the time of this writing the planned investigations of data storage and access methods, performance and scalability of the database, and the software development process, are all completed, or well underway.

[1] Harvey Newman (Caltech HEP), Julian Bunn, Juergen May, Les Robertson (CERN IT Division), Paul Messina (Caltech's Center for Advanced Computing Research) and Paul Bemis (then HP Chief Scientist)

The kernel of the GIOD system prototype is a large (˜ 1 Terabyte) Object database of ˜ 1,000,000 fully simulated LHC events. Using this database, we investigated scalability and clustering issues in order to understand the physics analysis performance of the database. Our tests included making replicas of portions of the database, by moving objects in the WAN, executing analysis and reconstruction tasks on servers remote from the database, and exploring schemes for speeding up the selection of small sub-samples of events. Other tests involved creating hundreds of ”client” processes that were simultaneously reading and/or writing to the database, in a manner similar to simultaneous use by hundreds of physicists, or in a data acquisition farm.

Current work includes the deployment and tests of a Terabyte-scale database at a few US universities and laboratories participating in the LHC program. In addition to providing a source of simulated events for evaluation of the design and discovery potential of the CMS experiment, the distributed database system is being used to explore and develop effective strategies for distributed data access and analysis at the LHC. These tests are using local, regional (CalREN-2, NTON) and the Internet-2 backbones in the USA, and the transAtlantic link to CERN, to explore how the distributed system will work, and which strategies are most effective.

2 Computing Infrastructure

In GIOD, we adopted several key technologies that seem likely to play significant roles in the LHC computing systems: OO software (C++ and Java), commercial OO database management systems (ODBMS; specifically Objectivity/DB), hierarchical storage management systems (specifically HPSS) and fast networks (ATM LAN and OC12 regional links). In detail, the project used several avant garde hardware and software systems:

- The Caltech HP Exemplar, a 256-PA8000 CPU SMP machine of ˜ 4600 SPECfp95
- The High Performance Storage System (HPSS) from IBM
- The Objectivity/DB Object Database Management System
- A Sun Enterprise 250 server with dual 450 MHz CPU, 512 MBytes RAM, attached Exabyte 230D DLT tape robot with dual drives, attached nStor FibreChannel RAID controller with ˜ 1 TByte disk in RAID0 and RAID5 arrays
- An HP-C200 Workstation equipped with a dedicated 155 MBits/sec optical fiber link to the Exemplar (via a FORE ATM switch)
- Pentium II-class PCs running Windows/NT, including an HP ”Kayak” with a special ”fx4” graphics card
- Various high speed LAN and WAN links (ATM/Ethernet)
- C++ and Java/Java3D

3 Database scalability and access

3.1 Initial scalability tests

We developed a simple scaling test application in C++ for an ODBMS, which we used to generate and select many thousands of objects in a pair of databases. We measured selection speed as a function of the number of objects in each database, the results showing that the fastest selection speeds were obtained by using indexes on the objects, and the slowest speeds were with text-based ”predicate” selections. We then measured the speeds on various hardware platforms and operating systems, ensuring that both databases were completely contained in the system caches, so that we could disregard effects due to disk access speeds. These results demonstrated the platform independence of the ODBMS and application software, and served to illustrate the performance differences due to the speeds of the CPUs, the code generated by the C++ compilers, and so on.

Other tests showed how the application and database could reside on different systems, and what impact on performance there was if they did: we measured the matching speed on a local database, and with a database stored on a remote server. For the problem sizes we were using, there was no significant performance degradation when the data were held remotely from the client application. We also measured the speed at which large numbers of databases could be created within a single ODBMS ”federation”. Our results for Objectivity/DB are shown in Figure 1, where a federation of $> 32,000$ databases was successfully created in about one week of continuous running.

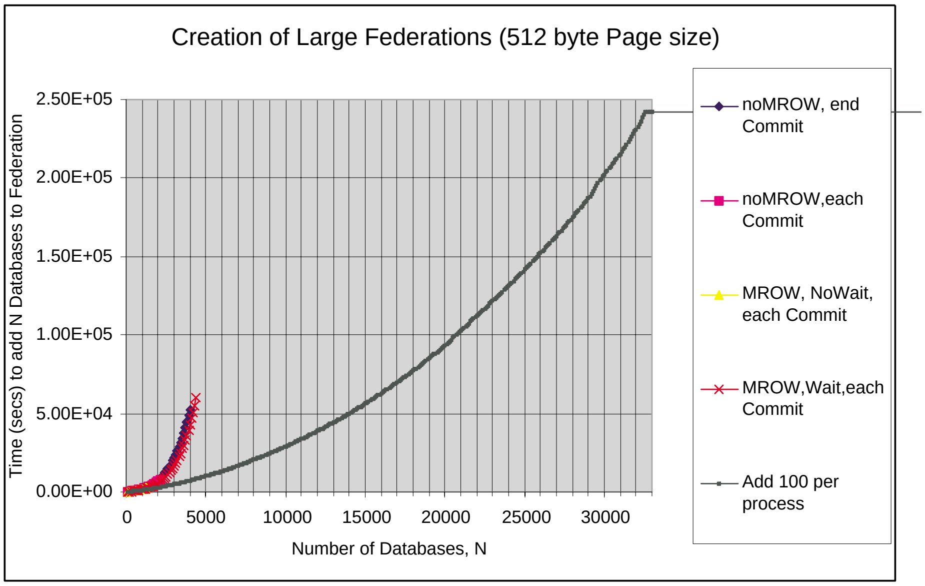

Figure 1: Showing the time taken to add new databases to an existing Objectivity federation. We were able to create a federation of 32,000 databases before getting too bored.

3.2 Summary of the simple Scalability Tests

These results demonstrated the platform independence of both the database and the application, and the locality independence of the application. We found, as expected, significance query/selection performance gains when objects in the database were indexed appropriately.

3.3 Extended scalability tests

These tests were performed on the HP Exemplar machine at Caltech. The Exemplar is a 256 CPU SMP machine of some 0.1 TIPS. There are 16 nodes (see Figure 2), which are connected by a special-purpose fast network called a CTI. 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 [3].

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

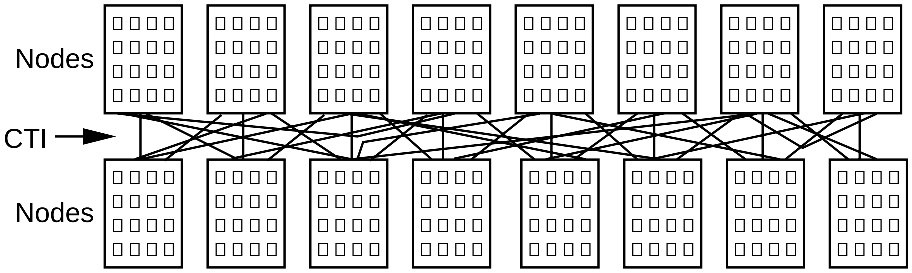

Figure 2: CPU/Node configuration of the HP Exemplar at Caltech

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 [4] as an interconnect.

3.3.1 Tests with synthetic HEP events

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.

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. 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 Holtman[6]. 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 catalogue and the journal files on a fifth file system. In reading the raw data, we used read-ahead optimisation.

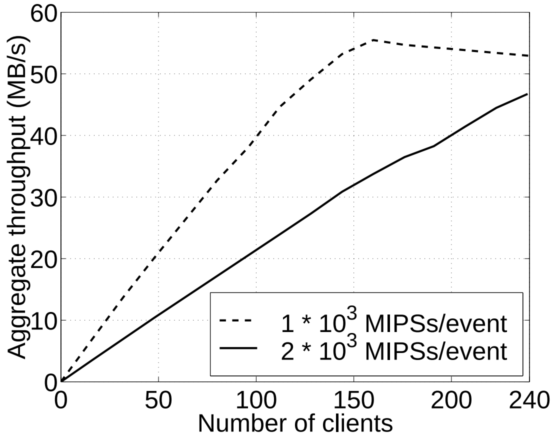

Figure 3: Scalability of reconstruction workloads

The results from our tests are shown in Figure 3. 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 3 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.

3.3.2 Tests with fully simulated CMS events

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 4 shows an example). 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.

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 catalogue was placed on a C200 HP workstation, connected to the Exemplar over a dedicated ATM link (155 Mbits/second). The clients accessed the catalogue over TCP/IP connections to the Objectivity/DB AMS server, which ran on the C200 workstation.

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 four disks that 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 the first test, the event model developed has a raw event consisting of 6 objects, with 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 are shown in Figure 5.
- In a second test we tried to quantify the overheads associated with the event data model described. These tests used a simpler ’1–container data model’, in which all 6 objects in the raw event were written to a single container, without object associations being created.

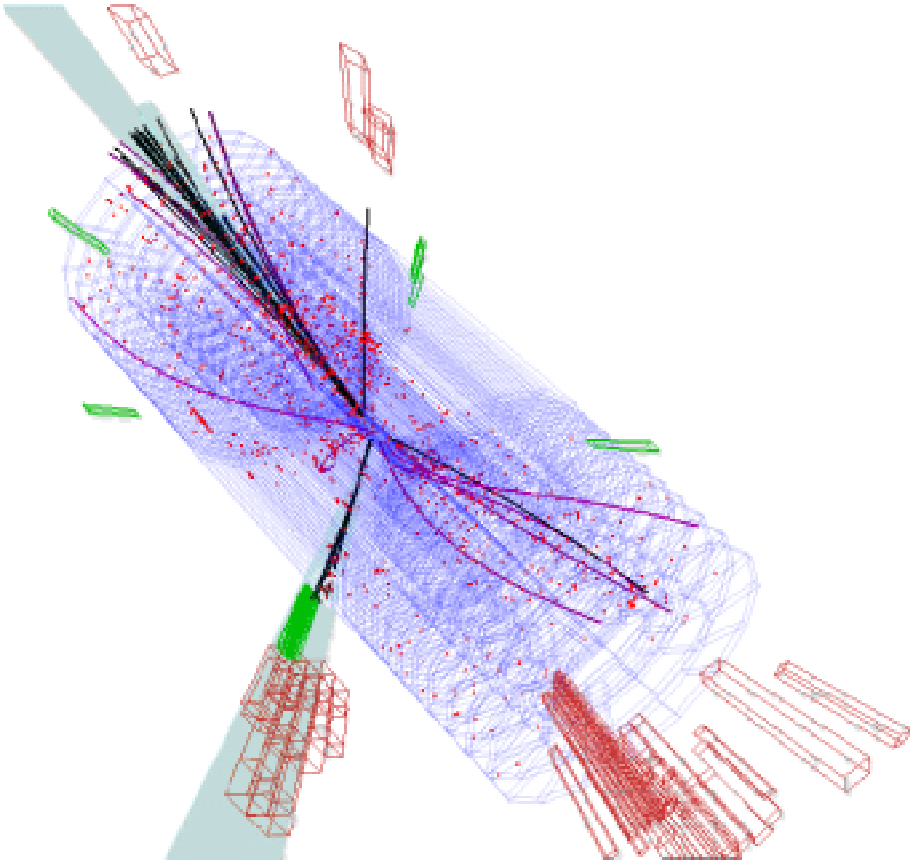

Figure 4: A typical multi-jet QCD event seen in the JavaCMS (Java2/Java3D) event display, with its tracks, detector space points and energy clusters

3.3.3 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 6 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 multi-container event data model a rate of 154 MBytes/second was achieved when running with 30 clients. We note that the overhead associated with this event structure is not significant.

In the earlier tests with synthetic data (section 3.3.1), the scaling curves flatten out, when more clients are added, because of a limits on the available disk bandwidth on the Exemplar. In the real physics data tests of Figure 6, the curves flatten out before the available disk bandwidth is saturated. In this case we found that an access bottleneck to the federation catalogue file was the limiting factor.

The federated catalogue file was located remotely on a C200 workstation connected to the Exemplar via an ATM link. A client needed to access the catalogue file whenever it created a new database, and whenever it deleted 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

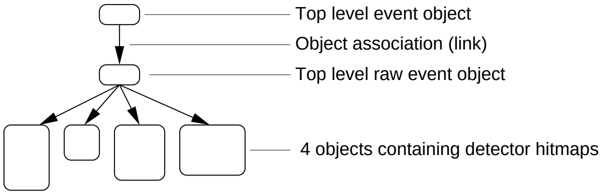
Top level event object
Object association (link)
Top level raw event object
4 objects containing detector hitmaps

Figure 5: Objects and their relations in the GIOD data model

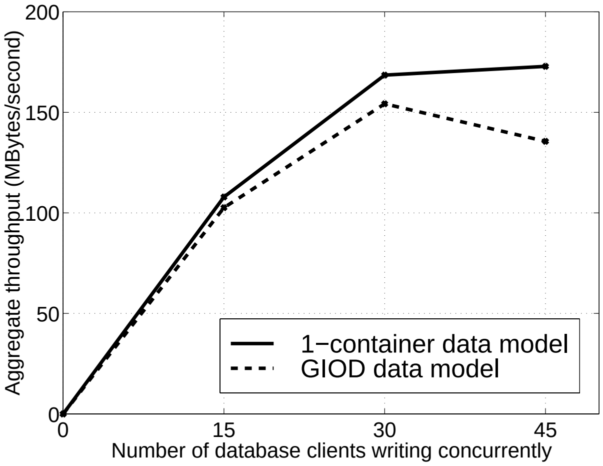

Figure 6: DAQ tests with real physics data

were busy waiting for their turn to access the remote catalogue file.

The cause of access bottleneck to the remote catalogue file was 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, 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 ˜ 10 Mbits/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 catalogue file, and updating only a few.

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

3.4 Conclusions from the Scalability tests

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 was excellent, with reasonable to good utilisation of the allocated disk resources. Note 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 catalogue were located remotely. In this configuration, the system achieved aggregate write rates into the database of more than 170 MBytes/second. This exceeded the 100 MBytes/second required by the DAQ systems of the two main LHC experiments, and satisfied a software and computing milestone of the CMS experiment.

We note that an obvious way to improve on the scaling limit is to create larger database files, or to place the catalogue file locally on the Exemplar system, as was done in the tests with synthetic data (Section 3.3.1). Another option is to create a large number of empty database files in advance.

We conclude that our measurements tend to confirm the viability of using commercial Object Database Management Systems for large scale particle physics data storage and analysis.

4 Other GIOD investigations

4.1 The Versant ODBMS

We evaluated the usability and performance of Versant ODBMS, Objectivity's main competitor. Based on these tests we concluded that Versant would offer an acceptable alternative solution to Objectivity, if required.

4.2 Object Database Replication between CERN and Caltech

We tested one aspect of the feasibility of wide area network (WAN)-based physics analysis by measuring replication performance between a database at CERN and one at Caltech. For these tests, an Objectivity/DB "Autonomous Partition" (AP) was created on a 2 GByte NTFS disk on one of the Pentium PCs at Caltech. This AP contained a replica of a database at CERN. An AP was also created at CERN with a replica of the same database. Then, an update of 2 kBytes was made every ten minutes to the master database at CERN, so causing the replicas in the APs to be updated. The transaction times for the local and remote replications were measured over the course of one day.

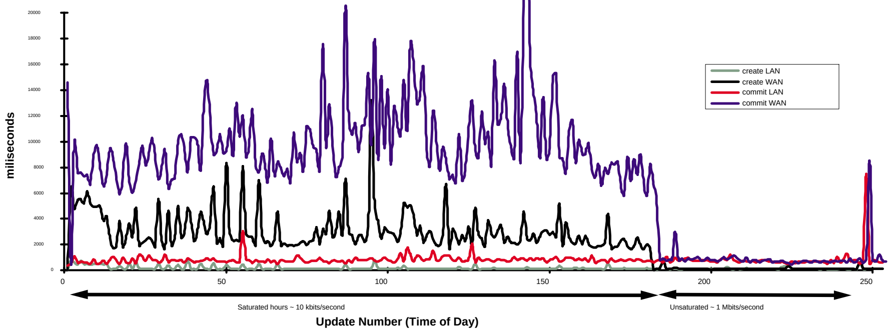

Figure 7: Time to achieve an update and commit it, as a function of time of day

The results (shown in Figure 7) showed that during "saturated hours" (when the WAN is busy) the time to commit the remote transaction is predictably longer than the time to commit the local transaction. On the other hand, when the WAN is quieter, the remote transaction takes no longer than the local transaction. This result demonstrates that, given enough bandwidth, databases may be transparently (and seamlessly) replicated between remote institutions.

4.3 CMSOO - Tracker, ECAL and HCAL software prototype

In 1998, CMS Physicists had produced several sub-detector orientated OO prototypes (e.g. for the Tracker, ECAL and HCAL sub-detectors). These codes were mainly written in C++, occasionally with some Fortran, but without persistent objects. We took these codes and integrated them into an overall structure, redesigning and restructuring them where necessary. We then added persistency to classes where it made sense by using the Objectivity/DB API. We reviewed the code and its structure for speed, performance and effectiveness of the algorithms, and we added some global reconstruction aspects. These included track/ECAL cluster matching, jet finding and event tagging. The final application, called CMSOO, was successfully ported and built for Solaris, HP/UX, and Windows/NT.

Concurrent to this work, Caltech/HEP submitted a successful proposal to NPACI (the U.S. National Partnership for Advanced Computing Infrastructure) that requested a time allocation on the Exemplar in order to generate ˜ 1,000,000 fully-simulated multi-jet QCD events. This allocation resulted in an accumulated total of ˜ 1 TBytes of data which was stored in Caltech/CACR's HPSS system. The events were used as a copious source of "raw" LHC data for processing by the CMSOO application and for population of the GIOD database.

The Java API supplied with Objectivity/DB proved to be an extremely convenient and powerful means of accessing event object data (created using CMSOO) in the GIOD database. We developed a 3D event viewer, which directly fetched the CMS detector geometry, raw data, reconstructed data, and analysis data, all as objects from the database. A screen capture from the event viewer has already been shown in Figure 4.

Working in Java again, we examined an early version of SLAC's Java Analysis Studio (JAS) software, which offers a set of histogramming and fitting widgets, as well as various foreign data interface modules (DIMs). Using JAS, we constructed a DIM for Objectivity/GIOD, and a simple di-Jet analysis routine. Using this analysis routine, we were able to iterate over all events in the CMSOO database, apply cuts, and produce a histogram of the di-jet mass spectrum for the surviving events.

Finally, we further explored the capabilities of the Java/Objectivity binding by developing a demonstration track fitting code. This code was able to efficiently find and fits tracks with $P_t > 1GeV$ in the CMS tracker. The good track identification rate was ˜ 1 per second, for a total of ˜ 3000 digitisings in the tracker. This compared favourably with the C++/Fortran Kalman Filter code we used in our production CMSOO reconstruction code.

4.4 Use of HPSS/NFS for Objectivity database files

We tested the operation of Objectivity/DB with a federated database located on an HPSS-managed NFS mounted file system. An application was run successfully against the disk-resident files. Then the database bitfiles were forced off HPSS disk and onto tape, and the application was again run. This caused an RPC timeout in the Objectivity application during the restore of the databases from tape to disk. After inserting a call to "ooRpcTimeout" in the application, specifying a longer wait time, we were able to re-run the application successfully.

4.5 WAN throughput and traffic profiles for Objectivity database clients

The more recent work in GIOD focussed on network-related aspects of using the TeraByte-scale CMSOO database. Some tests involved distributing a number of database client processes on the Exemplar, and having them communicate with a database hosted remotely via the dedicated ATM fiber link on our HP C200 workstation, and via the WAN to machines at FermiLab and at San Diego Supercomputer Center. We measured the I/O throughput to the disk containing the database files, the I/O traffic on the WAN/ATM networks, and the load on the database host processor during the tests. One clear conclusion was that the Objectivity database lock and page servers play important roles in governing the maximum throughput of the system.

5 Summary

The GIOD project work has resulted in the construction of a large set of fully simulated events, used to create large OO database federations, an activity that has brought useful practical experience with a variety of associated problems. Prototype reconstruction and analysis codes that work with persistent objects in the database were developed. Facilities and database federations were deployed as testbeds for Computing Model studies in the MONARC Project. Full details on the project are available at http://pcbunn.cacr.caltech.edu/.

The GIOD project has proved to be an excellent vehicle for raising awareness of the computing challenges of the next generation of particle physics experiments, both within the HEP community, and outside in other scientific and network-aware communities. For example, the JavaCMS event viewer was a featured demonstration at the Internet-2 Fall Meeting 1998 in San Francisco. At the SuperComputing '98 conference, the event viewer was demonstrated at the HP, iGrid, NCSA and CACR stands. The project was a featured potential application at the Internet2 Distributed Storage Infrastructure workshop at Chapel Hill in March 1999. GIOD was represented at the EDUCAUSE conference at Long Beach in October 1999, and at the SuperComputing '99 show in Portland in November 1999. These events have strengthened our relationship with organisations like Internet-2 and NTON, and have helped pave the way to CERN becoming a fully-fledged Internet-2 member late in 1999.

6 Acknowledgements

We would like to acknowledge the following individuals for their help and collaboration during the project: James Amundson (FNAL), Eva Arderiu-Ribera (CERN), Greg Astfalk (Hewlett Packard), Josh Bao (Caltech), Saskya Byerly (Caltech), Shane Davison (Objectivity), Manuel Delfino (CERN), Vincenzo Innocente (CERN), Juergen May (CERN), Reagan Moore (SDSC), Shahzad Muzaffer (FNAL), James Patton (Caltech), Ruth Pordes (FNAL), Les Robertson (CERN), Sergey Shevchenko (Caltech), Jamie Shiers (CERN), Christoph von Praun (ex. CERN) and Roy Williams (Caltech).

References

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

2 Objectivity/DB. Vendor homepage: http://www.objy.com/

3 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

4 Myrinet network products. Vendor homepage: http://www.myri.com/

5 TOPS, Testbed for Objectivity Performance and Scalability, V1.0. Available from http://home.cern.ch/˜kholtman/

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