Critical Database Technologies for High Energy Physics

David M. Malon
Decision and Information Sciences Division
Argonne National Laboratory
Argonne, IL 60439
malon@anl.gov

Edward N. May
High Energy Physics Division
Argonne National Laboratory
Argonne, IL 60439
may@anl.gov

Abstract

A number of large-scale high energy physics experiments loom on the horizon, several of which will generate many petabytes of scientific data annually. A variety of exploratory projects are underway within the physics computing community to investigate approaches to managing this data. There are conflicting views of this massive data problem:

- there is far too much data to manage effectively within a genuine database;
- there is far too much data to manage effectively without a genuine database;

and many people hold both views. The purpose of this paper is to begin a dialog between the computational physics and very large database communities on such problems, and to stimulate research in directions that will be of benefit to both groups. This paper will attempt to outline the nature and scope of these massive data problems, survey several of the approaches being explored by the physics community, and suggest areas in which high energy physicists hope to look to the database community for assistance.

1 Introduction

When planning for the now-defunct Superconducting Super Collider began several years ago, its anticipated data rate—on the order of one petabyte (1,000 terabytes) of data per experiment per year—seemed extraordinary. Now the ATLAS collaboration at the Large Hadron Collider at CERN expects to operate at a multipetabyte data scale, and several other experiments (at the Stanford Linear Accelerator, at Brookhaven National Laboratory, at the Thomas Jefferson National Accelerator Facility) will have roughly comparable data management needs. The calculation is straightforward—even at a sampling rate of only 100 hertz and sample sizes of 1 megabyte, just 10^7 operational seconds per year will yield a petabyte of data. With data collected over a 5- to 10-year period (and 2-4 concurrent experiments per accelerator facility), data samples in the tens of petabytes are expected to be the norm by 2010.

1.1 Background

High energy physics experiments have always generated large amounts of data. Traditionally, raw data have been refined in a series of event reconstruction and analysis steps to produce a sequence of successively smaller datasets used by the bulk of a collaborations' physicists.

A simplified view of the process is this. In high energy physics experiments, particle collisions occur at a rate far too high to allow data capture from all of them. In such a context, an "event" is, roughly, that which causes the data acquisition system to save a data sample. Out of many millions of particle collisions per second (40 megahertz at ATLAS), hardware "triggers"

The submitted manuscript has been authored by a contractor of the U.S. Government under contract No. W-31-109-Eng-38. Accordingly, the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.

Proceedings of the 23rd VLDB Conference
Athens, Greece, 1997

cull the candidates to some tens of thousands of potentially interesting events per second (75 kilohertz at ATLAS). A second-level software trigger further filters the sample to about 1 kilohertz. Another, more computationally demanding software filter is then applied; approximately 100 events per second will survive this third-level trigger. For each of these events, approximately 1 megabyte of data will be saved and passed to a series of reconstruction and analysis steps, which infer and calculate the identities and properties of subatomic particles present in the event, fit tracks, and more. ATLAS estimates that 100 KB of event summary data, 10 KB of analysis object data, and 100 bytes of event tag data will be generated per event by this process, in addition to the 1 MB of raw event data.

In earlier experiments with lower input rates, such derived data have been organized into compressed, self-describing data structures stored in flat files, and accessed by means of specialized I/O packages that can be linked to physicists' analysis codes. Data have been organized variously into row-wise (by event) and column-wise (by attribute) n-tuples, placed on parallel data servers, and connected to integrated physics analysis, statistics, and visualization packages.

Because it has never been feasible to store all data online, raw data, and data high in the analysis chain, are typically stored on tape. *A priori* judgments of what constitutes an interesting event, and what data from a given event are of greatest relevance, have dictated the physical organization and placement of data. This physical organization not only reflects, but also shapes, physicists' queries—questions whose answers require mounting thousands of tapes, in general, go unasked.

1.2 Current Investigations

In preparation for the next generation of experiments, physicists have begun to explore alternative approaches to this massive data problem. Their code development has become increasingly object-oriented, and their approaches to data management have followed the same path. Data modeling efforts for a number of physics experiments have concluded that object models describe experimental physics data quite well.

A research and development project at CERN (RD45) proposes a very large federation of Objectivity databases—current plans suggest 10,000 100-gigabyte databases per petabyte. When a query requires access to a database that happens to be on tape, the database would be imported in its entirety. Smart physical database organization would make it possible to keep the most frequently accessed data on many terabytes of disk.

The Computing for Analysis Project (CAP) at Fermilab, with somewhat nearer-term needs, has explored a different approach. Data are divided into "primary" and "secondary" classes in a persistent object store. In the current experimental implementation, primary data are partitioned across 8 nodes of an IBM SP PowerParallel system. Secondary data reside on tape. User queries run concurrently on 16 nodes of the same SP system, connected to the 8 data server nodes (and to each other) through a fast switch. Queries select events of interest on the basis of primary data only. For the events thus selected, requests for the corresponding secondary data are batched with those of other users so that tape access can be optimized. Data from this step are returned to the user in a format understood by current-generation physics codes (largely FORTRAN) for additional analysis.

The PASS project at Argonne National Laboratory has developed and implemented a flexible, lightweight object persistence manager that provides

- transparent access to every persistent object from every query node, no matter where in secondary or tertiary storage it may reside;
- support for efficient reorganization of data at the segment level, including striping and reclustering, without knowledge of object schemata;
- extensible support for a variety of storage mechanisms, including local and remote disk, raw RAID, Unitree file systems, raw device access to DD2 and 8 mm tape, parallel file systems such as IBM's Vesta and PIOFS, and Internet data access via standard FTP and HTTP mechanisms or cgi-bin scripts;
- support for data replication;
- support for multiple access paths to data;
- portability to heterogeneous distributed architectures.

The system has been used to store data from the Fermilab D0 experiment, and ISAJET simulation data for ATLAS. The system is intended to provide a tool to investigate approaches to data organization and clustering, caching and migration, replication, multiple data access paths, nonuniform data access and multilevel storage, and parallelism, and to understand the roles of parallel file systems, mass storage architectures, and non-dedicated parallel computing platforms. In order not to needlessly inhibit concurrent use of (and eventual migration to) commercial object-oriented databases, the user interface is compliant with a substantial subset of the C++binding in the Object

Database Management Group's ODMG-93 Version 1.2 specification. The PASS project has also investigated CORBA[1]-based approaches to distributed data access, including an internal implementation of a subset of the Object Management Group's Persistent Object Service Specification, and a study of the Object Query Service Specification.

In planning for the next generation of experiments, physicists often think in terms of a multi-tier data environment, with massive amounts of data at the experimental site, terabytes of data cached or replicated at regional centers (e.g., one in Europe, one in North America, one in Japan), and gigabyte samples on individual physicists' workstations. They aspire to a seamless environment in which, from a physicist's point of view, the system looks the same whether she/he is querying a 10 MB sample on a desktop workstation or a 10 TB sample on a massively parallel processor–the query and the physics code's interface to the data should be identical (and, ideally, the physicist might not even be aware of where a query is running).

For the ATLAS experiment, this distributed collaborative environment will involve approximately 1700 physicists at hundreds of institutes worldwide, with 500 people involved in data analysis, and 150 of those using the primary data system concurrently.

As of this writing, a new collaborative initiative involving a number of national laboratories and universities across the United States to develop a common approach to large-scale data handling for high energy and nuclear physics has been approved for funding as a scientific Grand Challenge by the U.S. Department of Energy. Because of the near-term time scale of some of the participating nuclear physics experiments (STAR and PHENIX at Brookhaven National Laboratory's Relativistic Heavy Ion Collider), the project's initial emphasis is on efficient physical clustering, caching, and management of multilevel and distributed storage to deliver bits to applications as efficiently as possible, with an ODMG-compliant data model integrated into a CORBA-aware analysis framework.

2 Special Problem Characteristics

While the scale of these problems is daunting, a number of factors serve to make data management and analysis simpler than in more general database settings; unfortunately, a number of other special problem characteristics conspire to make matters more challenging.

- Events are essentially independent of one another (or conditionally independent given experimental run conditions). This means that many queries may be parallelized almost arbitrarily in principle, with each processor operating independently on a disjoint subset of a very large collection of events.

- Access patterns are write-seldom, read often. (One does not alter experimental data, but beyond the writing done in adding 10 terabytes of new data per day, occasional updates are made, for example, to account for recalibrations, better track-fitting algorithms, and so on.)

- Queries are computationally intensive, and not readily expressible in standard query languages (SQL-x, OQL), unless those languages allow invocation of user-provided code. An example might be, "select all events E that contain at least two oppositely charged muons whose pairwise mass is greater than C, and for which myIntensiveComputation(E) yields TRUE." It may be easy to select all events with oppositely charged muons via the query language, and a query language expert might even be able to iterate over all distinct pairs of muons in the event to compute pairwise mass, but at some point, the partial selection must be turned over to user code.

- Collections may be very large, commonly on the order of 10^9 elements. The number of objects of a given type—the cardinality of an *extent*—may exceed the range of 32-bit integers by several orders of magnitude.

- Complex queries may take months to complete (or seconds—the range is enormous).

3 Needs and Wants

The following is a brief wish list, gleaned from several experiments, of features the physics data community would like to see in database products.

- Address at least tens—eventually, hundreds—of petabytes of data.

- Support collections of 10^9 or more elements efficiently.

- Support hundreds of simultaneous queries, some requiring seconds, some requiring months to complete.

- Support addition of 10 terabytes of data per day without making the system unavailable to queriers.

- Return partial results of queries in progress, and provide interactive query refinement.

[1] Common Object Request Broker Architecture

- Provide query cost estimates. (At least, warn if a query will mount a thousand tapes.)

- Manage data on both secondary and tertiary storage, and interact intelligently with emerging hierarchical mass storage subsystem architectures for resource scheduling, query optimization, and data caching and migration.

- Optimize tertiary storage access for simultaneous queries. Determine the data needs of concurrent queries and sort (or have the storage system sort) their tape retrievals into ordered per tape requests. (Apart from serious I/O speed degradation, tapes are not random access devices–they break under such utilization.)

- Support data access and analysis over an international wide-area network.

- Support invocation of user code from the query language, and/or allow piping of events selected by an OQL/SQL query-in-progress into user code.

- Provide a very flexible object model, with support for schema evolution and versioning.

- Provide flexible facilities for physical storage management and optimization and object data reclustering.

- Support statistical selection mechanisms (uniform random samples with and without replacement, query language support for quantile-based selection on indexed attributes without iterating through the entire collection, and so on). These are not uniquely problems of scale.

- Provide a data analysis environment that is identical on desktop workstations and centralized data repositories.

- Provide straightforward means to "check out" samples from the primary database and operate on them offline (from the perspective of the primary database), with implicit support for smart deep-copy operations.

- Take scalability issues into account in the development of database standards.

The last point requires explanation. A critical review of such welcome efforts as the Object Database Management Group's ODMG-93 Version 1.2 specification is beyond the scope of this paper, but there are several areas in which the current specification (in its C++binding) poses potential scalability concerns. Examples include lifetimes of object references (too long to accommodate very large-scale data access), iterators that must be bidirectional, even for unordered collections, insufficiently articulated support for data clustering, and transaction execution models that inhibit large-scale data access and concurrency.

3.1 Some Comments on Tertiary Storage Management

Tertiary storage management introduces a number of challenges. If my query touches 1% of a collection of 10^9 events, and data are cached and clustered so effectively that 99% of the relevant events reside on disk, how many tapes will I need to mount in order to retrieve the final 1% of my sample (1% of 1% of the total data—10^5 events)? Under realistic assumptions about tape capacity and event size, the answer is (almost) all of them, with probability (almost) one.

The same problems arise with multidimensional indexing schemes. If data are indexed and clustered so effectively in anticipation of my query that 99.9% of the data I want reside contiguously, either on disk or on a small number of tapes, the remaining 0.1% will require mounting essentially every tape anyway.

While the hope is that systems may be tuned so that satisfying most queries does not require touching petabytes of data, querying the entire petabyte-scale database should nonetheless be possible for data mining agents and for physicists willing to incur the wait.

3.2 Some Comments on Parallelism

It is clear that any system capable of supporting queries against petabytes of data must take advantage of parallel processing. Such parallelism should be transparent to users, who should not need to care whether their queries are running on one processor or a thousand. Efficient parallelism, though, may require addressing a variety of interface issues. It may be important, for example, to provide collections and iterators that support nondeterminism in the order in which elements are returned; without this, there are serialization problems that significantly impede scalability, or else queries must run essentially in batch mode, with a substantial sorting task remaining as a postprocessing step. A physicist iterating over an unordered collection (or possibly even an ordered collection whose order is not germane to the query) should receive data as fast as possible, and the underlying system should support the potential for nondeterminism–which 1,000 events are returned first may vary from run to run depending on such factors as processor workload, and this is perfectly acceptable. (Related concerns about the ODMG-93 requirement of Standard Template Library-style bidirectionality in its iterators were noted above.) Support for queries of the form, "return ANY 1,000 events satisfying..." is also desirable.

Collection constructs and their implementations may be designed with a particular emphasis on scalability and potential for parallelism. Some approaches, such as ParSets, which explicitly declare the disjoint nature of set contents, have appeared in recent database literature, although there may be some static dependence on how data are allocated. Other high-level approaches are also possible. For example, the ODMG specification allows one to say A is the union of B and C, but this is an assignment, not a definition—if one later adds an element to B, it does not become an element of A. Collection constructs that allow A to be *defined* as the union of B and C enhance scalability. (Imagine a collection AllEvents, and what may happen to queries if, in a transaction-based architecture, an exclusive lock must be acquired on AllEvents every time an event is added to the database.)

There are many subtleties involved in efficient parallelism, and an example of the potential for subtle dependence on how databases are populated might be in order here. A physicist might run a large-scale simulation by replicating it, with different random number seeds, on each of P parallel processors. It is likely that, to achieve reasonable scalability, these processes will attempt to fill disjoint pieces of persistent storage. The effective consequence may be physical clustering into at least P clusters, where P is entirely an artifact–an accident of the number of processors available when the simulation is run. That artifact will, however, persist in the physical layout of the database, and affect optimal storage access for subsequent queries, no matter which or how many processors those queries may utilize.

4 Conclusion

There is a vast difference between being able to store and retrieve massive amounts of data, and managing such data effectively. At petabyte scales, where the storage problem alone is daunting, physicists hope the database community will provide some of the tools necessary to derive new knowledge and understanding from the large-scale physics experiments of tomorrow.

References

[BIN95] Pavel Binko, Dirk Duellmann, Jamie Shiers, "CERN RD45 status report—a persistent object manager for HEP," *Proceedings of the International Conference on Computing in High Energy Physics '95,* World Scientific, 1996, pp 334–338.

[FID95] K. Fidler et al, "The Computing for Analysis Project—a high performance physics analysis system," *Proceedings of the International Conference on Computing in High Energy Physics '95,* World Scientific, 1996, pp 304–308.

[MAL95a] David Malon et al, "Object database standards, persistence specifications, and physics data," *Proceedings of the International Conference on Computing in High Energy Physics '95,* World Scientific, 1996, pp 319–323.

[MAL95b] David Malon et al, "Data analysis in an object request broker environment," *Proceedings of the International Conference on Computing in High Energy Physics '95,* World Scientific, 1996, pp 329–333.

[MAL97] David Malon and Edward May, "An ODMG-compatible testbed architecture for scalable management and analysis of physics data," *Proceedings of the International Conference on Computing in High Energy Physics '97,* to appear April 1997.

[CAT96] R.G.G. Cattell et al, *The Object Database Standard: ODMG-93 Release 1.2* (Morgan Kaufmann, San Francisco, 1996).

[OBJ93] Object Management Group, *The Common Object Request Broker: Architecture and Specification, Revision 1.2* (OMG, Draft 29 December 1993).

[SIE94] Jon Siegel et al, *Persistent Object Service Specification,* OMG Document Numbers 94-1-1 and 94-10-7 (Object Management Group, 1994).

[IBM95] IBM et al, *Joint Submission: Object Query Service Specification,* OMG TC Document 95-1-1 (Object Management Group, 1995).

[DEW94] D.J. DeWitt, J.F. Naughton, J.C. Schafer, S. Venkataraman, "ParSets for parallelizing OODBMS traversals: implementation and performance," *Computer Sciences Department Technical Report, University of Wisconsin, Madison,* 1994.