CONF-930461--3

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.

ANL-HEP-CP-92-101

Requirements for a System to Analyze HEP Events Using Database Computing*

E. May, D. Lifka, E. Lusk, L. E. Price
Argonne National Laboratory

C. T. Day, S. Loken, J. F. MacFarlane
Lawrence Berkeley Laboratory

ANL-HEP-CP--92-101

DE93 009960

A. Baden
Department of Physics
University of Maryland

R. Grossman, X. Qin
Department of Mathematics, Statistics, & Compute Science
University of Illinois at Chicago

L. Cormell, A. Gauthier, P. Leibold, J. Marstaller, U. Nixdorf, B. Scipioni
Superconducting Supercollider Laboratory

MAR 26 1993
OSTI

We describe the requirements for the design and prototyping of an object-oriented database designed to analyze data in high energy physics. Our goal is to satisfy the data processing and analysis needs of a generic high energy physics experiment to be proposed for the Superconducting SuperCollider (SSC), and requires the collection and analysis of between 10 and 100 million sets of vectors (events), each approximately one megabyte in length. We sketch how this analysis would proceed using an object-oriented database which supports the basic data types used in HEP.

Introduction

In this paper, we describe the requirements and a conceptual system for analyzing high energy physics (HEP) data using database computing. This section and the following three sections are adapted from [10] and [13]. The system is designed to scale up to the size required for high energy physics experiments at the Superconducting SuperCollider (SSC) laboratory. These experiments will require collecting and analyzing approximately 10 to 100 million "events" per year during proton colliding beam collisions. Each "event" consists of a set of vectors with a total length of approximately one megabyte. This represents an increase of approximately 2-3 orders of magnitude in the amount of data accumulated by present large HEP experiments.

Interactions between protons in the counter-rotating beams at the SuperConducting SuperCollider will occur at a rate of approximately 10^8 Hz. The collider is expected to go on-line during 1999. A detector trigger system will select between 10-100 Hz of these collisions (or events) for writing to permanent storage for further analysis. An average event will contain approximately one Megabyte of information. This large quantity of data - up to one thousand Terabytes per year, assuming 10^7 seconds of actual beam collisions per year - along with an expected one

*This work supported by the U.S. Department of Energy, High Energy Physics Division under Contract No. W-31-109-ENG-38.

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

thousand physicists distributed at approximately one hundred different sites requires a means of accessing the data that is a scale-up of approximately three to four orders of magnitude over the largest existing high energy physics experiment. The storage and access requirements are unlikely to be met by projected advances in commercial database systems.

The project goal is to find a data model, and storage, access, and analysis methods which will meet present requirements for HEP data storage, handling, and analysis and be scalable to the 10^4 increases required by the end of the decade at the SSC. We believe this to be a generic problem in both the public and private sector in planning for the distribution and analysis of increasingly large data sets. At the SSC, this will be especially true.

Traditional data analysis in HEP

At present, the CDF colliding beam experiment at the Fermi National Accelerator Laboratory (FNAL) measures the radiation products from the collision of particle beams at rates of up to 285 kHz. Collisions, or events, are recorded in detectors which provide ~150 kbytes of digital information. However, only a small fraction (~ 1 Hz) of the total number of events are recorded on magnetic tape for data analysis. The computing environment can be broken down into the following categories:

- **Online**: In the *online* stage, events which warrant scientific investigation are identified through a sequence of several decisions (a mixture of hardware and software processing), each requiring greater execution time. A decision to keep a particular event, called a *trigger*, results in the corresponding raw data being collected and written to magnetic tape.

- **Production**: In the *production* stage, the raw data is processed by computer codes (production codes) which reconstruct the digital data words in order to identify the number and type of the particles which characterize each event. Specifically, the result of the production codes are dimensional quantities such as energy, mass, momentum, orbits, event topology, and *etc.* These data summaries are written to some medium, usually magnetic tape, for further analysis, referred to as *data summary tapes* (DSTs).

- **Analysis**: In the *analysis stage*, computations are performed on the data in the data summary tape. For instance, such computations can be used to discover (or verify) correlations between certain types of events, or measure an average of a quantity over a particular set of events.

Most HEP computing schemes store data in sequential records using memory management software systems written at HEP laboratories. Data are analyzed via computer programs which are written mostly in Fortran, access events from storage as records, read it into common memory, and provide hooks for users to apply custom-written subprograms. Each of these programs are compiled and linked to a standard set of libraries. Linking, at present, is not usually done interactively.

Once the analysis program is built, it is made to loop over a number of events. For each event the entire record is read from storage, and those For each event the entire record is read from storage, and those parts which are relevant to the particular analysis (and very few analyses use all of the data in a single event) are used by the user subprogram. For instance, a typical session would consist of:

1. Read in each event. If the events are on tape, access the tape first. For each event:
   - require certain global event characteristics;
   - calculate quantities from each event (*e.g.* the presence of a certain number of a certain type of particles);
   - require the particles to have certain characteristics;

- fill histograms or scatter plots of distributions.

2. If desired, save events passing requirements onto disk or tape.
3. Report results of calculations, histograms, plots, regressions, *etc.* at the end of session.

Although this procedure is straightforward, inefficiencies arise since the procedure is repeated several times to produce sequentially smaller datasets for analysis by different working groups. Also, although it is often convenient to produce additional datasets, this is not usually done, due to the expense and difficulty of doing so. The critical point is that access to the data is accomplished via entire event records, and in a sequential manner, *i.e.* events are read as atomic.

Expectations at the SSC are that there will be an increase relative to the largest ongoing HEP experiments to date of about ~1 order of magnitude in the size of each event (to 1 Mbyte/event), ~1-2 orders of magnitude in the number of events collected for analysis (from 1 to 10-100 Hz to tape), and ~1 order of magnitude in the number of physicists who will be accessing the data (~1000 physicists).

Examples of current HEP analysis tools

Recently, the data analysis stage has been made easier by using an interactive data analysis program called PAW [1] and a data access package called Adamo [2], both of which were developed at the European Organization for Nuclear Research (CERN) in Geneva, Switzerland. The program PAW (Physics Analysis on Workstations) resembles a windowing environment and incorporates an inline Fortran interpreter. The Fortran interpreter eliminates the need to need to link the large Fortran programs previously used to analyze the data. The interpreter has been found to be fast enough, since most of the time spent when running in the interpreter mode is for I/O. Data analysis with PAW must be preceded by a transformation of the data to a limited relational tuple. However, once the data is transformed, the program allows one to interactively identify events with computed quantities passing certain thresholds *cuts* and to study correlations and anti-correlations of the particles and events. The program also supports many standard statistical operations, such as constructing histograms, scatter plots, curve of best fit, and *etc.* However, the system has well-defined data structures which do not allow dynamic changes, and require a preprocessing to transform existing HEP data (which is by nature dynamic).

Adamo is a data structuring package, developed for the Aleph experiment at CERN, and written in Fortran. It is based on relational tables and allows the modeling of the complex event structures used in HEP. However, its modeling is limited to a single event at a time, that is, a complete event is read into memory and restructured into Adamo tables. The program can then access portions of the event through relational queries. While this model is good for production code, where all information about a single event is needed at once and information about other events is irrelevant, it is not appropriate for analysis, which involves the collection of statistics from a large number of events, each statistic based on only a subset of each event's information.

Scientific Databases

A database provides access to data via queries regarding the data itself, and not its physical or logical location. The relational model is a simple and powerful model suited to many business and commercial applications, especially those applications making use of simple data in a fixed format. Applications, such as computer aided design, computer aided manufacturing, office information systems, and artificial intelligence, have resulted in extensions to the relational model [3], and new models, such as the object-oriented data model [5]. More recently, applications requiring the management of scientific and engineering data, such as time series, satellite data, DNA sequence data, seismic data, and collider physics data have re-

sulted in attempts to define a data model suitable for these types of applications [4], [6], [8] and [9]. Just as there is currently no agreed upon definition of an object-oriented data model, there is also no agreed upon definition of a data model for scientific computations.

Issues which turned out to be important in our system are somewhat different than issues of importance in business or CAD/CAM applications. For our application in HEP, the main requirements of our system are:

- the ability to query very large amounts of data (expected to greater than 1PB), stored in a hierarchical storage system [7]
- the ability to efficiently execute numerically intensive queries specified by Fortran or C functions
- the ability to support large, complex objects
- a mechanism for working with *derived data*, that is data that is derived from other data via some computation, and the ability to dynamically determine which data should be precomputed or computed in the background

Issues of importance in other scientific databases [4]. such as the level of interpretation of the data, the intended analysis of the data, the source of the data, and the use of metadata do not present major problems in this application.

Data Modeling

Before any issues of the physical realization and access methods to large data stores supporting a scientific data base, the issue of an appropriate data model must be addressed. We believe the entity-relationship model is not sufficient to the task, that an object-oriented scientific data model is required. It should have the following notions:

objects The basic entities are objects. Objects are instances of classes. Examples of classes include Integers, Strings, Vectors, FourVectors, Events, and Particle Candidates. The notion of abstract data typing or encapsulation must be supported.

attributes Objects have internal states or attributes.

methods Objects have functions or methods acting on them.

inheritance Classes can inherit attributes and methods from other classes.

collections Objects may be grouped together into collections. Typically, this is done by selecting objects from other collections on the basis of their attributes or on the basis of the results of methods acting on them.

derived data A class C is said to be *derived* from one or more other classes D_i, if the values of instances of class D are functionally dependent on instances of the classes C_i. When the values of instances of class D_i are changed, the changes are propagated (either automatically, or after certain triggers) so that instances of class C are recomputed.

versioning Multiple versions of objects, with histories and time interval validity must be supported.

Some typical HEP class and object hierarchies are shown in Fig. 1 the (a)underling physics, (b)the detector hardware, and (c)event reconstruction process. Fig 2. shows a dynamically derived object (resonant state) from an object hierarchy relationship.

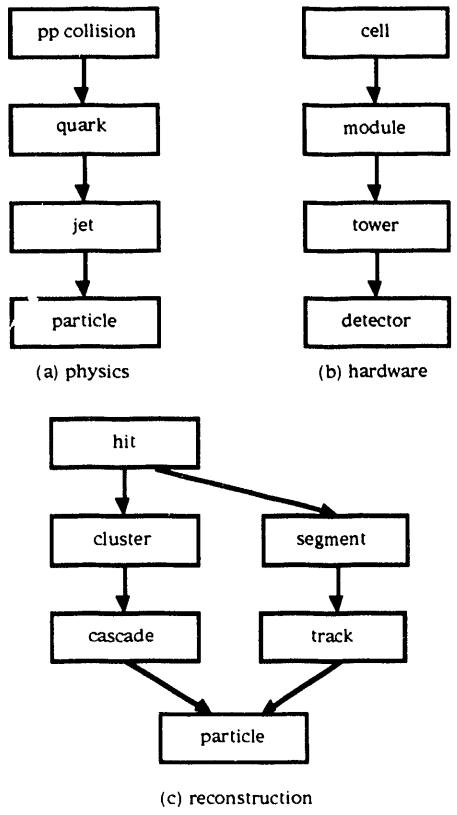

Figure 1: Typical HEP Class and Object Hierarchies

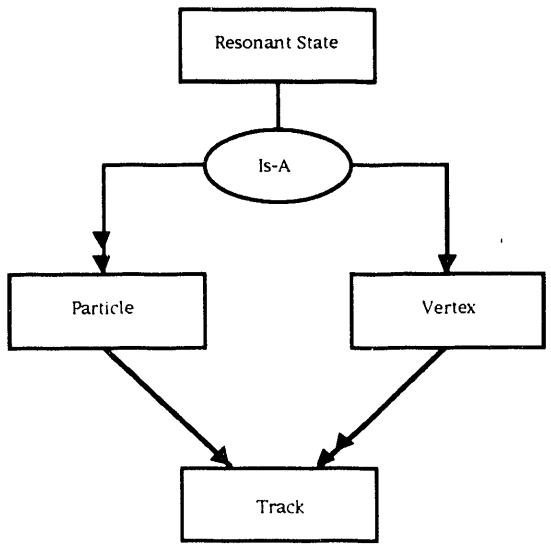

Figure 2: Complex derived object hierarchy relationship.

Design Considerations

There is an inherent "chunkiness" to HEP data since each event is largely independent of the previous and following events. This allows very simple parallel processing model for much of the compute intensive analysis tasks. Queries in a data base representation which do not correlate objects in one event with another should also parallelize very efficiently.

The tree structure of objects in an event can reach depths of four or more. In simple relational designs, leaves of the tree need unique keys obtained by concatenating keys from all the layers above them. This can lead to considerable storage overhead in very large databases. Furthermore, piecing together an event by doing joins at run time could lead to the time for a single event growing with the size of the entire database.

For object databases, keys are replaced by fixed size pointers, regardless of depth of the event structure. Furthermore, run time joins are not needed, so the overall scaling behavior may be better than relational systems. However, the scale up for SSC is so enormous that more study is needed for both systems.

As analysis proceeds, events often have new objects attached to them. For relational systems, one just defines a new table with the proper keys; all the old data is undisturbed. For object systems, however, if the event has a fixed set of pointers to its component objects, adding a objects could lead to a complete restructuring of the database, a prohibitively expensive proposition. A Lisp-like list structure may be more stable and is under investigation.

A conceptual model for realizing an HEP event store and database

A conceptual realization is shown in Fig. 3. The HEP computer user sees the computing system primarily via a workstation (WS) as a Data (object and/or event) Store. The workstation is a client either on a local LAN or a remote LAN connected to the Data Store via a wide area network. Queries generated by the workstation are examined by

the local DBMS to see if they can be satisfied by data (mostly objects) stored on the local WS disks. Those which can not are passed to the local (master/sibling meta-data base). The results (events and objects) are made available on the local file systems as cached data for the workstation to analyze. User queries which can not be satisfied by the local data based caches or the metadata caches are analyzed and optimized for transmission to the main event/object store. This is a set of shared memory parallel computers (nP) each with typically a 100TB tape robot, a 100 GB RAID disk array (for storing methods, objects and events) and interconnected by a high speed LAN to the meta database and the workstations. These IO cluster processors will be optimized for moving data from tape-to-disk-to-network. The incoming queries from the workstation users will trigger stored methods for the events and/or objects in the IO clusters which are expected to reduce the returned events and/or objects which satisfy the user queries such that the outgoing data flow is consistent with high speed network capacity.

This type of architecture demands a number of requirements.

- A data base management system which works well in a client-server hierarchical mass storage system (e.g. IEEE mass storage reference model).
- I/O subsystems which provide high capacity, low latency and high throughput tape robots.
- A Data Model which exploits the inherent parallelism in of HEP data, which will allow a linear speed up in both computing resources and data access.
- A Query language which refers naturally to physics objects and can express complex and numerically intensive queries.

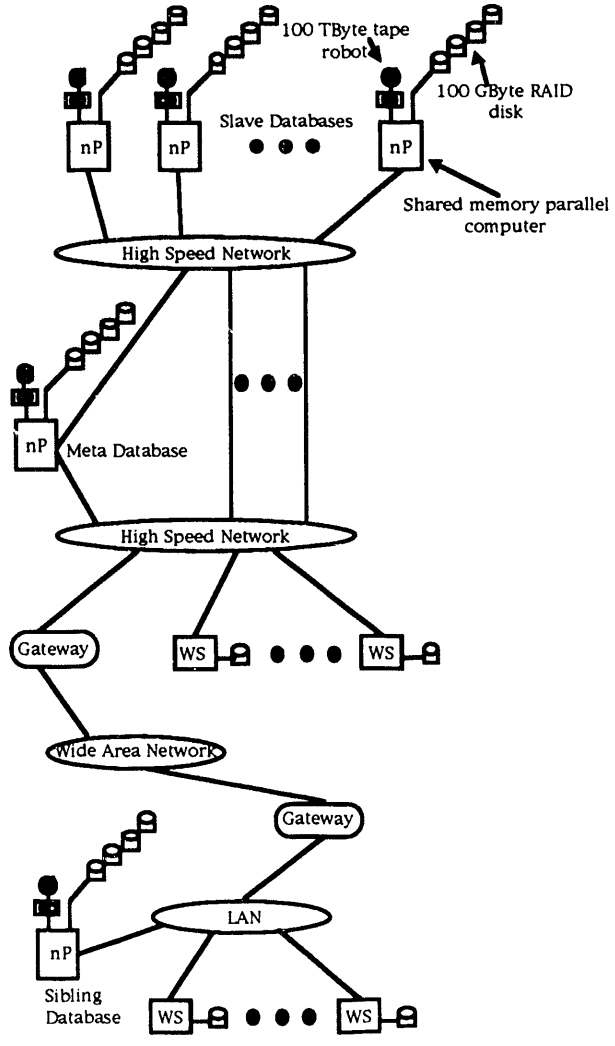

Figure 3: Conceptual Implementation

Outlook

We believe the computing industry will provide cost effective IO sub-systems for realizing the mass storage hardware requirements. We believe and encourage the development of the IEEE mass storage model for the integration of these IO sub-systems with a network based computing model. We believe the design and development of database management software systems which can efficiently exploit such a computing model is a challenging given the magnitude to the data store to be accessed. The authors of this paper are pursuing preliminary designs and prototypes for a system suitable for HEP analysis as described in references [10] and [11].

A project known as PASS [12] has begun work on solving the problems outlined here on a time scale useful to SSC detectors. Funded by funds from the DOE part of the High Performance Computing and Communications initiative in addition to program funds from the DOE office of High Energy and Nuclear Physics and of the SSC, we have begun a 3-phase attack on the problem. We have the three stages of this work over a 5 year period as:

Phase 1. The goal of Phase 1 is to provide a proof of concept for analyzing HEP data using database computing. This phase began in May, 1991 and is expected to be completed by June, 1992.

Phase 2. The goal of Phase 2 is to build an experimental prototype (the Mark II system), demonstrating the critical technologies required to build a full scale system. This phase is expected to last two years.

Phase 3. The goal of Phase 3 is to build a working prototype (the Mark III system), that can be migrated into a full scale production system. This phase is expected to last two years.

The PASS project will try to prototype new versions of a system appropriate for database computing in HEP. Rather than build an entire system from the ground up, the project will try to make use of commercial technology whenever appropriate. For this reason, the system is being designed in a layered and modular fashion. Emphasis is also placed on those parts of the design which are considered critical paths and which the market does not seem to be large enough to justify the entry of a commercial vendor. Rather than try to build a production strength system, our interest is in pushing the analysis and software technology far enough to solve the problem, and then interesting others (commercial partners) in taking our designs and prototypes; then developing commercial or production strength versions.

Acknowledgements

Work supported by the U.S. Department of Energy, Division of High Energy Physics, contract W-31-109-ENG-38.

References

[1] M. Goosens (Editor), "PAW -Physics Analysis Workstation," *Cern Program Library entry Q121*, CERN, Geneva, CH 1989, unpublished report.

[2] S. M. Fisher and P. Palazzi, "The ADAMO Data System," *CERN/ECP/PT xx/92*, CERN, Geneva, CH 1992, unpublished report.

[3] M. Carey and L. Haas, "Extensible database management systems," *ACM SIGMOD Record*, Volume 19, 1991, pp. 54-60.

[4] J. C. French, A. K. Jones, and J. L. Pfaltz, "Summary of the final report of the NSF workshop on scientific database management," *ACM SIGMOD Record*, Volume 19, 1991, pp. 32-40.

[5] W. Kim, "Object-oriented databases: definition and research directions," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 2, 1990, pp. 327-341.

[6] V. M. Markowitz and A. Shoshani "Representing object structures in relational databases: A modular approach," *Lawrence Berkeley Laboratory Technical Report*, No. LBL 28482, January 1991.

[7] "Mass Storage System Reference Model, Version 4" edited by Sam Coleman and Steve Miller, IEEE. to appear.

[8] Maurizio Rafanelli, "Research Topics in Statistical and Scientific Database Management," in *Statistical and Scientific Database Management*, M. Rafanelli, J. C. Klensin, and P. Svensson, editors, Springer-Verlag, Berlin, 1989.

[9] A. Shoshani, "Properties of statistical and scientific databases," *Lawrence Berkeley Laboratory Technical Report*, No. LBL 29900, November, 1990.

[10] R. Grossman, A. Baden, C. Day, D. Lifka, E. Lusk, E. May, and L. Price, "Analyzing High Energy Physics Data Using Database Computing: Preliminary Report," *Laboratory for Advanced Computing Technical Report, Number LAC91-R17*, University of Illinois at Chicago, December, 1991.

[11] R. Grossman and X. Qin, "PTool: A Software Tool for Working with Persistent Data", *Laboratory for Advanced Computing Technical Report Number 92-11*, University of Illinois at Chicago, 1992. To appear.

[12] E. May, R. Lusk, L. Price, D. Baden, R. Grossman, C. T. Day, J. F. MacFarlane, "Database Computing for the SSC", *Proposal to the DOE HPCC Initiative*, Argonne National Lab, 1991, unpublished.

[13] D. Baden, C. T. Day, R. Grossman, D. Lifka, E. Lusk, E. May, S. Mehta, L. Price, X. Qin, "A Data Model for Scientific Databases Supporting Objects, Attributes, Methods, and Collections: Preliminary Report", *Laboratory for Advanced Computing Technical Report*, University of Illinois at Chicago, December, 1991

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

END

DATE FILMED

6/8/93