Title
DATA MANAGEMENT FOR HIGH ENERGY PHYSICS EXPERIMENTS PRELIMINARY PROPOSALS

Author
Olken, F.

Publication Date
1987

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA, BERKELEY

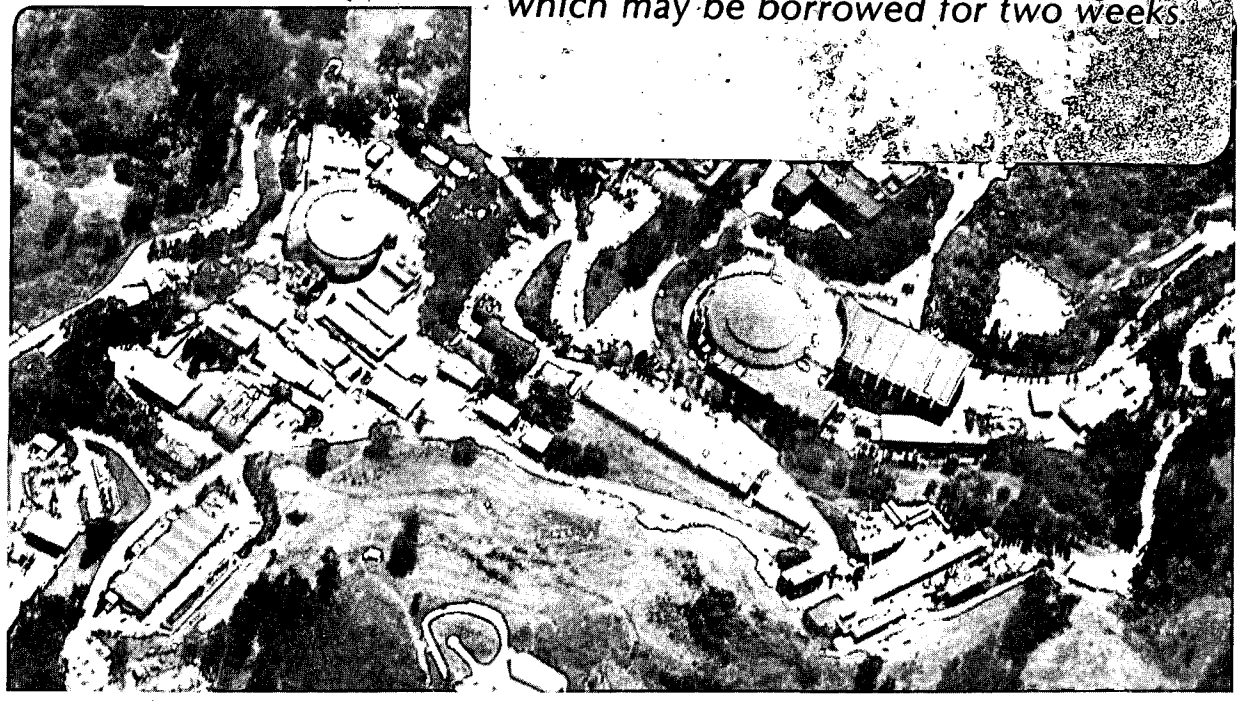
Information and Computing Sciences Division

Presented at the Computing in High Energy Physics Conference, Pacific Grove, CA, February 2-6, 1987, and to be published in the Proceedings

DATA MANAGEMENT FOR HIGH ENERGY PHYSICS EXPERIMENTS PRELIMINARY PROPOSALS

F. Olken, S.C. Loken, D. Rotem,
A. Shoshani, and T.G. Trippe

January 1987



Prepared for the U.S. Department of Energy under Contract DE-AC03-76SF00098

DISCLAIMER

LBL-22883

Data Management for High Energy Physics Experiments
Preliminary Proposals

Frank Olken, Stewart C. Loken, Doron Rotem,
Arie Shoshani, and Thomas G. Trippe

Computer Science Research Department
Lawrence Berkeley Laboratory
University of California
Berkeley, California 94720

January, 1987

This research was supported by the Applied Mathematics Sciences Research Program of the Office of Energy Research, U.S. Department of Energy under contract DE-AC03-76SF00098.

Data Management for High Energy Physics Experiments - Preliminary Proposals *

Frank Olken, Stewart C. Loken, Doron Rotem,
Arie Shoshani, Thomas G. Trippe

Lawrence Berkeley Laboratory
Berkeley, CA 94720

Abstract

Currently HEP experimental data are reduced as they become available. We propose instead a "demand driven" approach to data analysis. Full analysis will be performed only as needed, in response to user queries which specify the subset of events for which reduced data is needed.

To support this approach we propose to partition the datasets on the cross product of several trigger inputs, instead of storing the data in chronological order. Queries will be automatically decomposed into a set of requests against several partitions. Indexing, physically clustering the data on the logical partitions, and caching of partitions will be employed for efficiency.

1 Introduction

Current practice in the analysis of high energy physics experimental data is to store the data in chronological order as they are produced from the detectors on magnetic tapes. Track reconstruction is then performed on all of the recorded data. These *reduced data* are also kept in chronological order. Finally, users sequentially search this data set to extract subsets of interest for further analysis. These subsets are typically specified in terms of ranges of several event parameters.

We propose a different approach to the analysis and management of HEP data. Our efforts are motivated by several considerations. First, much of the present track reconstruction effort is wasted because many of the events are never used. It is estimated that approximately 10^8 instructions will be required to reconstruct a single event for the DØ experiment. Second, the sequential searching of the entire dataset is very slow for such large datasets. Third, the emergence of optical disk technology as viable storage medium permits block random access to data. Fourth, there have been developments in database management techniques which are relevant to this problem.

Our approach entails several ideas:

1. "Demand driven" data analysis so that track reconstruction is only performed when needed.

2. Indexing event attributes for faster query processing.

3. Physical clustering of similar types of

*Issued as LBL tech report LBL-22883, January 1987. This work was supported by the Applied Mathematical Sciences Research Program and by the Office of High Energy and Nuclear Physics, High Energy Physics Division, both of the Office of Energy Research, U.S. Department of Energy under Contract DE-AC03-76SF00098. Electronic mail address: olken@lbl-csam.arpa

events to reduce volume mounts required to access target subsets.

4. Automatic subset management: reuse results of previous queries to answer new ones.

5. Automatic storage hierarchy management: caching recently used data on magnetic disk for faster access.

6. Automatic query optimization, to exploit the various indices, subsets, etc. in formulating query procesing plans.

Next, we briefly provide a schematic description of the HEP datasets and give a sample query. We then proceed to discuss the first four topics. The last two topics are omitted due to space constraints, see [3].

2 HEP Datasets

We include here a brief discussion of the various types of HEP experimental data to facilitate the subsequent discussion of physical database design.

There is a variety of HEP experimental data: raw detector data, level 2 trigger data (i.e., the results of real-time analysis), analyzed data (e.g. track reconstruction), and summary data. The analyzed data consist of individual particle tracks, jets, and data concerning entire events. Summary data here refers to aggregations (e.g. histograms, χ^2 values) over groups of events. In addition to the data concerning events, there are also calibration data concerning the state of the DØ detector. Space limitations preclude any discussion of summary data in this paper.

The level 2 trigger data are *precomputed* and recorded with the raw data in the database. The results of the track reconstruction (variously referred to as *analyzed or derived attributes*) will be computed only when needed, and then saved for reuse.

Part of the database "schema" will be a dependency graph describing the relationships among various derived event attributes, the software modules, and the input data required to compute each derived attribute. This will be used to support the demand driven computation of derived event attributes as described in Section 3.

The raw detector and level 2 trigger data comprise some 10^7 events per year of operation for the DØ detector. Each event consists of about 10^5 bytes of raw and level 2 trigger data. Hence the raw data for a year occupy about 10^{12} bytes. The analyzed data are expected to account for a similar volume of data.

2.1 A Sample Query

Following we show a query which we believe to be representative of the anticipated workload: retrieve event attributes for events with at least one electromagnetic jet greater than a standard energy threshold and a total electrogmagnetic energy greater than 30 GeV and an effective mass between 60 and 90 GeV/c^2, or more formally, retrieve event attributes for events with $n_{J_{EM}} \geq 1$ and $E_T(EM) > 30\ GeV$ and $60\ GeV/c^2 < m_T < 90\ GeV/c^2$.

Two of the simple predicates are half-range queries (a simple inequality), one is a ordinary range query (two sided inequality). This type of query is considered a *partial range query* since only 3 out of many possible attributes are specified. The first two attributes specified, $n_{J_{EM}}$ and $E_T(EM)$, are usually available as level 2 trigger input data in an experiment like DØ, i.e., at the time the event data are written to tape. The third attribute, m_T, would typically only be available after the events are analyzed. Below we discuss the implications of this on query processing.

3 Demand Driven Analysis

Current practice consists of sequentially logging "interesting" event data on tape, and then fully analyzing (track reconstructing) the entire raw dataset.

We envision "demand driven" analyses. The raw data will be physically partitioned onto various volumes so that "similar" types

of events are stored together on the same volume (tape or optical disk).

Users will then specify which set of events they want analyzed and which attributes are needed.

The system will employ various indices to reduce the number of events which must be examined to determine if they qualify for selection.

Once a set of candidate events has been identified, the system will check to see if the desired event attributes (and any further selection attributes) have been computed for these events. For those events which have not had all the required derived attributes computed, the raw data will be retrieved and the required attributes computed.

Given an incompletely analyzed event, and a list of required derived attributes, the system will use the dependency schema to automatically determine which software modules must be invoked, what antecedent and intermediate data are required, and then construct a computation schedule for the event. Since many events will have similar analysis histories, caching of computation plans will expedite this process. After the new data are computed, a bit vector describing which data have been computed for this event will be updated and the new data recorded in the event record.

4 Indexing

In order to avoid examining all of the recorded events for each query a variety of indices will be used.

The primary index will dictate the physical layout of the data. We expect that the primary index will consist of a multi-dimensional index built out of some of the level 2 trigger data inputs. This is discussed further in Section 5.

Because it is impractical to construct a primary index on all of the level 2 trigger attributes, we expect to construct secondary indices on some of the level 2 trigger data inputs and perhaps trigger outputs. Since these items are available for all events, these can be full indices.

Both the primary index, and all of the secondary indices will be event level indices (i.e., they will resolve down to the level of individual events).

Secondary indices on derived attributes will be partial indices and include entries only for those events for which the particular derived attributes have been computed (usually a small subset of the entire set of events).

Such partial indices have two uses: to determine whether events qualify for selection, and to determine whether they have already been analyzed.

5 Clustering

As noted above, the primary index is the index which induces the actual physical layout of the data on the storage media. This will be a multi-dimensional index on some of the level 2 trigger data input terms [1]. Thus events will be partitioned by the primary index into groups with some similar event attributes. Each such group of similar events will be stored together on a tape or optical disk.

There has been considerable debate among the authors as to whether the primary index should be built from the trigger inputs or the trigger outputs. Candidate trigger input attributes are shown in Table 1. Trigger outputs are boolean combinations of equality, or inequality predicates on discretized versions of these attributes. The rationale for using trigger data (either inputs or outputs) for the primary index is that it is available at the time the raw data are logged to tape, and that it is of interest for selecting events. Using the trigger outputs to partition the data produces a partitioning that is quite well suited to answer queries which are similar to the triggers. However, using the trigger inputs produces a more versatile index and data partitioning, which while not as well suited to answer queries specified by the triggers, can

Symbol	Name	Range	Partitions
$\not{p}_T$	missing transverse momentum	0-255+ GeV/c	4
$E_T(EM)$	total energy of electromagnetic particles	0-255+ GeV	4
$E_T(all)$	total energy of all particles	0-255+ GeV	4
z_{vtx}	vertex location	0-112 cm	2
$n_{J_{EM}}$	EM energy thresholded jet multiplicity	0,1,2,3,4+	2
$n_{J_{tot}}$	total energy thesholded jet multiplicity	0,1,2,3,4+	2
n_μ	muon count	0,1,2,3,4+	2
$\rho_{TRD,J_{EM}}$	TRD-EM Jet spatial correlation	boolean	2
	Total number of primary partitions		2^{11}

Table 1: Table of Possible Primary Index Terms

more readily answer other queries, in effect changing the triggers.

Shown in Table 1 is are most likely candidates for the primary index terms.

By grouping similar events together, we can reduce the number of volumes (disks, or tapes) which must be mounted (and scanned in the case of tapes) to retrieve a particular subset of data.

Multi-dimensional indices (such as envisioned here for the primary index (and hence partitioning of the raw data)) do not perform very well if the number of attributes specified in a query is much less than that specified in the index. For partial match queries on KD-trees, KD-tries, or grid files, the expected access cost is at least $O(n^{(1-(s/k))})$, where n is the total number of records, s is the number of attributes specified in the query, and k is the number of attributes in the index [2].

Hence, in practice, preference must be given to some event attributes over others in clustering the data. Queries which select subsets on the basis of the clustering attributes will have much faster retrieval than those which specify nonclustered attributes.

We believe that a *KD-trie* [4] offers a suitable type of organization for the primary index. We begin by decomposing our search key into digits of radix r. The key will be used to specify a path through a tree composed of nodes, each of which contains r pointers (or nulls). The $k'th$ digit of the key is used to determine the corresponding pointer in the $k'th$ level node. This trie structure is simple and offers high fan-out and fast addressing. A KD-trie is a multi-dimensional trie in which digits of several attributes are interleaved to construct the trie. KD-tries are well suited to multi-dimensional range queries.

6 Data Reorganization

The events come out of the detector in chronological order, not clustered on event attributes. Hence it will be necessary to reorganize the data according to the desired clustering. This is basically equivalent to sorting the data into bins.

In the case of tapes the number of passes will equal to $\log_b(T)$, where T is the number of tapes (approximately 10^4 for D$\emptyset$) and b is the number of tapes which can be buffered on disk. Thus if we can hold 10 tapes on disk (about 10^9 bytes), four passes will be required to reorganize the data. The first pass can be done as the data are produced from the detector.

Because optical disks permit random access, we can reorganize the data in one pass provided that we have adequate disk buffering. This is desirable since the optical media are not reusable and are fairly expensive. The required disk buffering (assuming two optical disk drives) is $Dt_w(t_{mount})$ where D is

the number of optical disks to store the data (about 10^3 for DØ), t_w is the transfer rate at which data is being written (i.e., the output rate from the detector) (approx. 10^5 bytes/sec for DØ), and t_{mount} is the time to spin down a platter, unmount it, return it to the jukebox, retrieve a new platter, and spin it up - approx. 10 seconds. This amounts to 10^9 bytes of disk buffers for DØ.

7 Automatic Subset Management

Present practice typically consists of storing user specified subsets as separate datasets on tape. Usually both raw and analyzed data are stored together. Subset management is entirely manual, and subsequent reuse of the subset is dependent on human interaction and intervention.

We envision that much of the subset management will be automated, with machine processable specification of subset selection criteria, subset indexing, and automatic searching for relevant subsets when queries are processed. Such seraching can be done by employing theorem proving techniques to identify subsets whose selection predicates cover (include) the target selection predicate. This will permit commonly used subsets to be reused automatically. There has been some research (theory and small prototypes) on this in the data management community, but no practical experience as yet.

We plan to store analyzed data separately from raw data. Subsets will variously be stored as event ID lists, bit vectors over event ID domains, or physically instantiated as either complete analyzed records or partial records on either magnetic disk, tape or optical disk. Subsets will automatically be converted to more compact forms if they are unused for a period of time. Users will be permitted to advise the system of special storage preferences.

Data management research has suggested that not only the final results of user queries, but also intermediate results may be useful for subsequent query processing.

8 Conclusions

We believe that our approach offers significant reduction in the processing time and effort required to analyze HEP datasets. This approach is most appropriate when optical disks are employed as the storage media for the major datasets. For a more extensive discussion and bibliography of the data management techniques see [3].

References

[1] S. Aronson et al. *Design Report, The DØ Experiment at the Fermilab Antiproton-Proton Collider.* Technical Report, Fermilab, November 1984.

[2] Philippe Flajolet and Claude Puech. Partial match retrieval of multidimensional data. *Journal of the ACM*, 33(2):371–407, April 1986.

[3] Frank Olken. Physical database support for scientific and and statistical database management. In R. Cubitt et al., editors, *Proceedings of the Third International Workshop on Statistical and Scientific Database Management*, pages 44–60, EUROSTAT, Luxembourg, July 1986. Also issued as LBL-19940.

[4] Jack A. Orenstein. Multi-dimensional tries used for associative searching. *Information Processing Letters*, 14(4):150–157, June 1982.

LAWRENCE BERKELEY LABORATORY
TECHNICAL INFORMATION DEPARTMENT
UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA 94720