

PARALLEL QUERY PROCESSING FOR EVENT STORE DATA *

D. Malon, D. Lifka, E. May

Decision and Information Sciences, Mathematics and Computer Science,
and High Energy Physics Divisions,
Argonne National Laboratory
Argonne, IL 60439 USA

R. Grossman, X. Qin, W. Xu

Laboratory for Advanced Computing and
Department of Mathematics, Statistics, and Computer Science
University of Illinois at Chicago
Chicago, IL 60607 USA

FERMILAB

JUL 1 1994

LIBRARY

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-94-31



Abstract

Enormous data volumes and large, geographically dispersed user communities characterize the next generation of experiments in high energy physics and other scientific disciplines. Parallel processing will be integral to the solution of the information storage and retrieval problems that these experiments will engender.

We describe several approaches to parallel query processing that have been implemented in the early stages of the PASS (Petabyte Access and Storage Solutions) project. These have been tested on an object-oriented persistent event store built from Fermilab CDF data, and evaluated on the 128-processor IBM SP-1 at Argonne National Laboratory, as well as on networks of workstations.

INTRODUCTION

Responding to queries directed to petabyte-scale scientific databases in large multiuser environments will require significant parallel processing capabilities.

*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.

The nature of high energy physics data makes effective parallelism possible at a number of levels—individual queries may be parallelized, data servers and file systems may be parallelized, and even straightforward multiuser parallelism is relatively free of degradation due to lock contention since the preponderance of data are accessed by most users in read-only fashion. The following sections describe ongoing research in the PASS project in the area of query parallelization for large object-oriented scientific databases.

The focus of our parallel query processing work to date has been upon replicated query strategies, and upon parallel strategies for caching and migration of physically contiguous units of persistent storage, known as folios, in which collections of persistent objects reside.

REPLICATED QUERY STRATEGIES

Replicated query strategies follow the SPMD (single program, multiple data) model of parallel programming. Such a model applies when typical queries have the form "for each event that satisfies criterion A, return the derived data produced by computation B." Such queries can be parallelized readily by replication into queries against disjoint database subsets if events are essentially independent, or if events can be partitioned into essentially independent collections.

Two replicated query strategies have been studied in various guises. In the first, queries are sent to worker processes, who each satisfy the query against their local data and send results back to the master process. In the second, a workload queue is constructed, and workers in turn remove the first workload chunk that they can handle (e.g., for which they have access to the requisite database folios). When a worker completes a chunk of work, it sends results back to the master, and checks the queue for another chunk of work for which it has the resources. The process continues until the queue is empty, or until no worker can handle any of the remaining workload chunks.

PARALLEL FOLIO SERVERS

Work has also been undertaken on a different approach, in which references to

nonlocal data generate requests to a folio manager, who arranges delivery of the physical unit of persistent storage (folio) in which the desired object resides from one or more of a collection of parallel data servers.

IMPLEMENTATIONS

Trial implementations of these approaches have been tested on the Argonne and Fermilab IBM SP-1 PowerParallel systems, and on heterogeneous networks of UNIX workstations. The underlying database was built from Fermilab CDF data, and follows an object-oriented model formulated as part of the PASS project. An Argonne-enhanced version of PTool64, developed at the University of Illinois at Chicago, was used as the persistence manager. Interprocess communication in the parallel query tests was implemented using the Argonne-developed p4 package. To date we have developed an FTP-based implementation of the parallel folio server strategy, and a mechanism to move database folios directly over socket connections. We have also used IBM's Vesta parallel file system on the Fermilab SP-1 to parallelize delivery of database folios to single-user queries.

COMPARISONS

These approaches and their various implementations differ in both theoretical and practical ways. Differences include single-user and multi-user speedup, whether user code, database code, or storage system interface code needs to be parallelized, potential for load balancing, adjudication of access to shared data, levels of data communication traffic, parallel work queue management, data caching,

and satisfying queries that cannot be subdivided into pieces that can be handled by a single node.

Both replicated query strategies are relatively easy to parallelize, and do not require parallelization of (possibly proprietary) database packages. In our test implementations, parallel queries were constructed by hand from serial queries written in C++, but with a modicum of control at an Object Query Language (OQL) interface, automatic parallelization should be possible in many cases. The first query replication strategy requires only the ability to initiate the parallel queries and to aggregate the returned data; it does, however, rely upon the assumption that parallel workers have access to disjoint portions of the database. The workload queue implementation correctly handles the problem of shared data and implicitly provides dynamic load balancing when data are in fact shared, but the workload queue introduces the potential for a serial bottleneck when the number of parallel processes is large. Such a bottleneck should be avoidable by making workload chunks sufficiently large.

Parallel folio server strategies rely on the ability of the database package to connect to parallel and high-performance data servers, and may result in high data traffic. Parallel speedups derive from parallel data paths to multiple queriers, parallelism implicit in the data server (e.g., the Vesta parallel file system), and the potential for parallel prefetching.

It is likely that in a very large database environment, queries will benefit from parallelism both in the query processing and in the data services. It is also likely that a hybrid strategy that

partitions the query workload between the querying node and the database host nodes may prove the most promising. We have done some preliminary theoretical performance analysis along these lines, and we plan to investigate such strategies in the coming year.

REFERENCES

1. R.G.G. Cattell, editor, *The Object Database Standard: ODMG-93*, Morgan Kaufmann Publishers, San Mateo, CA, 1994.
2. C.T. Day, R. Grossman, D. Malon, E. May, L.E. Price, D.R. Quarrie et al, "The PASS Project Architectural Model," Draft, January, 1994.
3. R. Grossman, X. Qin, "PTool: A Low Overhead, Scalable Object Manager," *Proceedings of SIGMOD 94*, to appear.
4. E. Lusk, R. Overbeek, et al, *Portable Programs for Parallel Processors*, Holt, Rinehart, and Winston, Inc., New York, 1987.