

Multiple Query Optimization by Cache-Aware Middleware Using Query Teamwork*

K. O’Gorman, D. Agrawal, A. El Abbadi
University of California, Santa Barbara
{kogorman, agrawal, amr}@ucsb.edu

Abstract

Queries with common sequences of disk accesses can make maximal use of a buffer pool. We developed a middleware to promote the necessary conditions in concurrent query streams, and achieved a speedup of 2.99 in executing a workload derived from the TCP-H benchmark.

1. Introduction

Multiple query optimization has long been studied in the context of batches of queries for which an optimal global execution plan is sought [1][2]. We consider the online version of the problem: concurrent independent streams of queries, which are served from a common queue. Certain workloads produce streams of queries instantiated from a library of templates. Accordingly, the queries may share features that can be exploited in optimization, especially shared sequences of accesses to secondary storage.

2. Middleware for Team Formation

We developed a theoretical model [4] that predicts, and we have shown in a commercial database, that common sequences of accesses to secondary storage become synchronized by the buffer pool if they begin within a critical time period, the *time horizon* of the buffer pool. The time horizon is a function of the size of the buffer pool, the rate of processing of requests to secondary storage, and the experienced cache hit-rate of the buffer pool, and tends to be tens of seconds. Queries sharing a common sequence and synchronized in this way make maximal use of the buffer pool, and we refer to such a group of queries as a *team*.

We created a middleware that serves a workload of concurrent query streams. The middleware queues requests with a scheduling policy that attempts to improve the chances of team formation. Each scheduling event consists of scheduling the query at the head of the queue, and perhaps some candidate team members from

elsewhere in the queue. We can independently vary limits on the number of such teams and the number of database server processes (concurrent queries).

In general, queries instantiated from different templates may or may not have shared access sequences; in fact sometimes the same is true of queries from the same template. We explored this pairwise behavior by testing representative instantiations of all pairs of templates from the TPC-H [3] benchmark workload. Our main experiment was to run the first 10 query streams concurrently, each with one query instantiated from each of the 22 query templates of the TPC-H workload. We found that just forming teams of queries from the same template gave an overall advantage, but that it was even better to allow teams to form from different templates, but require the templates to have good performance from the pairwise test. We achieved a speedup of 2.99 compared to running the 10 query streams on 10 dedicated database servers on a system with a single disk drive.

3. Conclusion

We have developed a way to improve the execution of concurrent asynchronous independent query streams, with the advantages of not requiring batch operation and without the modifications to the underlying database engine.

References

- [1] Timos K. Sellis. Multiple query optimization. *ACM Transactions on Database Systems*, 13(1):23-52, 1988
- [2] Chen and Dunham. Common subexpression processing in multiple-query processing. *TKDE* 10(3), 1998.
- [3] Transaction Processing Performance Council. *TPC-H Benchmark Specification*. Published at <http://www.tpc.org/hspec.html>.
- [4] K. O’Gorman, D. Agrawal, A. El Abbadi, *Multiple Query Optimization by Cache-Aware Middleware Using Query Teamwork*, Technical Report 2001-18, UCSB, <http://www.cs.ucsb.edu/research/trcs/2001-18.shtml>.

* This work was partially supported by NSF grants EIA-9818120, IIS-98-17432, EIA-9986057 and IIS-99-70700.