Project abstract

Deterministic highly-scalable concurrency control within database systems

Prior research on deterministic database systems has proposed pre-determining a serializable schedule of transactions prior to their execution. The pre-determined schedule is represented via an explicit dependency graph, which determines the order in which transactions must execute and effectively obviates the necessity for explicit locking.

Over the course of last semester's CPSC 490, I have shown that building a dependency graph by considering batches of transactions rather than single transactions independently from one another does lead to much more efficient and parallelizable schedule. Although the theoretical results from simulations are promising, they do not measure the overheads present within a read database system. This semester I have built a real database system to investigate the interplay between the overhead of handling batches and the improvement that follows from more parallelizable schedules. In particular, I have focused on creating a system that minimizes the overhead of batch creation and allows for multiple batch schedules to be created at parallel.

The implementational part of the work has focused on decreasing the contention resulting from communication among threads and a large number of threads processing batches in parallel. While the system has not been fully profiled as of the time of writing of this report, it is very possible that the throughputs obtained are at least as good as those of the traditional concurrency control systems and potentially even better. Current work on the project focuses on memory management and investigation into efficient garbage collection for the system. The full discussion of overheads and the solutions employed to decrease them will follow in the appropriate sections. The system has been written in around 10000 lines of C++ and is under active development. This report will describe the general architecture of the system, the results I have currently obtained and the future work.

*dna@yale.edu
†NetID: ss2755, stanislaw.swidwinski@yale.edu