Project proposal

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.

During last semester we 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 results leave us hopeful, we have not built a working database system which uses this graph creation approach. During this semester we will build 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, we will focus on the two types of overheads introduced in the system: the overhead of creating batch schedules and that of incorporating them into the global schedule. We will also aim to explore the possibility of creating multiple batch schedules concurrently to increase parallelizability of batch creation.

Within the implementation work we will be focusing on decreasing contention resulting from communication among threads. First of all, the scheduling algorithm proposed earlier must be modified to decrease the contention when execution threads obtain transactions. This will be done by assigning ownership over transactions to execution threads arbitrarily. Every execution thread will be able to "steal" transactions from other threads if those owned by it are all blocked and the blocking transactions are not being executed. The communication among all threads will be done via Compare and Swap instructions on fields owned by transactions. Consequently, we will allow the built system to use the standard two-phase locking concurrency control approach to compare with our system. We hope to show that our approach is more performant than two-phase locking under high contention and just as performant under low contention workloads. Moreover, we will attempt to separate and consequently quantify the contributions of the two main sources of system's performance increase: the better utilization of shared locks and the reordering of transactions.

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