Project Proposal
Plan-ahead concurrency control for highly scalable database systems

Concurrency control is the mechanism by which a database system allows multiple transactions to run simultaneously while ensuring the results are “serializable”, or equivalent to a serial ordering. Traditional methods of concurrency control typically fall into two categories: those which, like two-phase locking, prevent conflicts by dynamically ordering transactions at runtime, and those which, like optimistic concurrency control (OCC), check for conflicts after the transaction has run and abort if any are found. We aim to investigate a newer method, “plan-ahead concurrency control”, which determines a serializable schedule for the transactions before runtime, allowing far fewer concurrency controls during query execution and opening the door to a number of more efficient orderings.

The basic problem of arranging transactions for better concurrency bears many similarities to set-packing, a well-known NP-hard problem. Essentially, the concurrency control footprint of a transaction can be reduced to the set of records in the database it touches, with the constraint that any two transactions which modify the same record conflict with each other and cannot be executed simultaneously. Under this framework, building the maximally efficient schedule involves breaking this set down into large subsets of that can run without conflicts, i.e. their write-sets are disjoint.

While optimally solving set-packing in its basic form is well studied and known to take exponential time, this application comes with a number of complicating wrinkles. Most significantly, while set-packing assumes all sets must be disjoint, concurrency control distinguishes between write accesses, which must be exclusive, and read accesses, which can occur simultaneously with other reads without conflict. This difference gives us extra flexibility in our packings. On the other hand, the time constraints of running in a live database constrict the available options, as the overhead of performing lengthy calculations to produce the best schedule may eliminate the advantages of such an ordering. Even if it were possible to calculate a single optimal or near-optimal packing in a reasonable amount of time, our problem requires us to process every transaction eventually; if an algorithm gives large initial packings at the cost of requiring a larger number of total batches, it may not actually outperform another algorithm with more modest performance on the initial batch. This property may necessitate looking at other related problems, including graph coloring and independent set.

The aim of my work this semester is two fold. First, since prior work of Stanislaw Swidwinski has demonstrated the promise of a greedy algorithm for scheduling, I plan to compare this greedy approach the optimal solution theoretically and via simulation, and investigate alternative approximation algorithms if the current approach leaves significant room for improvement. Second, I plan to compare the pre-scheduled approach to other methods of concurrency control both theoretically and in practice. To conclude this work, we will produce a writeup of the results.