Cache Management in a Deterministic DBMS

Daniel Abadi, Advisor

September 21, 2011

1 Determinism in databases

The database research group at Yale has been hard at work on a new “deterministic” database technology. The difference between this new approach to database design and that of traditional databases lies chiefly in how concurrent transactions are dealt with. In traditional ACID database systems, the database can only promise that the concurrent transactions will complete in some fashion such that the ultimate state of the database is equivalent to what the state of the database would have been if the incoming transactions had completed in some serial order. There is no guarantee as to what properties that serial order might have, and there is no guarantee that subsequent runs with identical concurrent transactions will result in the database having a final state equivalent to the same serial ordering.

The paper by Thomson and Abadi, The Case for Determinism in Database Systems, goes into some of the detailed pros and cons of the new deterministic approach including more elegant replication and reduced contention from shorter lock holds in distributed systems. For the purposes of my project, however, the primary advantage of the deterministic ordering of transactions is that the DBMS will know, even before dispatching transactions to nodes, which resources will be required and in what order. Furthermore, since the design of the deterministic DBMS places a transaction sequencer on each node, so (at least for the set of concurrent transactions) the cache manager is totally prescient of what resources will be needed and at what time, at least in the single-partition transaction case.

2 Cache Manager

For my senior project, I will design and implement a cache management layer for the deterministic DBMS. The chief problems I will try to address are

1. Maximally Leveraging Prescience - The cache management layer should take full advantage of the fact that any incoming packet of concurrent transactions will be deterministically serialized. From this list of transactions, the cache manager should guarantee, with assumptions, that there will be no cache misses.
2. Consider Variations in Read and Writes - The cache management layer should be robust and should not incur excessive cache misses as the result of the variable access times to the underlying disk layer.

3. Utilize Existing Cache Algorithms - Inevitably, the DBMS will have to periodically write older less-used memory pages to disk to make room for upcoming reads. Thus between transaction batches, I will need to implement existing schemes, such as LRU, as well as benchmarks to gauge how well the schemes perform under different workloads.

3 Deliverables

By the end of the project, I hope to deliver a fully functional, fully documented cache management layer for the deterministic DBMS project. In the process, I hope to discover the best way to accomplish each of the three points listed above. In particular, I will examine the current inter-node messaging scheme so as to maximize the information that each node will have in advance of its query execution. Secondly, I will need to learn how to create the said robust disk layer. Finally, I will need to reconsider the various cache schemes in the context of my prescience. The cache schemes I end up implementing will have rigorous probabilistic models backing their validity.

4 References