Review of "On Optimistic Methods for Concurrency Control"

Nicolas Bonvin

February 7, 2008

Innovative ideas

- The authors propose two families of optimistic concurrency controls that do not use locking. They realized that locking is only necessary in the worst case. These methods may be better than the traditional locking methods for systems where transactions conflicts are really low such as query-dominant systems or very large tree-structured indexes and they can potentially improve dramatically concurrency while eliminating the locking overhead for lots of database applications.
- Behind their optimistic idea:
  - Reading is completely unrestricted; reading a value can never break the integrity. However, returning a value from a query is considered like a write.
  - Writing is severely restricted.
- The paper states the main disadvantages of locking:
  - locking maintenance has a large overhead
  - lack of general purpose deadlock-free locking protocols
  - low concurrency in case of congested locking
  - locking is actually needed in very rare cases
- Sets of objects (read set, write set) accessed during a transaction are maintained by the concurrency control functions.
- There are 3 phases: read phase, validation phase, and write phase. During a read all writes are made on local copies. Validation ensures the integrity between the transactions. During the write phase local copies are written in the "global" database.
- Serial equivalence: individual transactions are serial equivalent if there is a serial sequence of the transactions that generate the final structure. This is a sufficient but not necessary condition for integrity. This is validated through the use of a transaction number.

Most glaring problems

- The authors don't provide experiments or realistic data to establish that conflicts are so rare. I think there may be several cases in realistic workload where "hot spots" exist in the database, sot that the probability of conflict increases.
- The solution proposed in this paper could be very expensive if the amount of changes is high: how to handle the different sets if they are bigger than the physical memory ?
- I'm not sure if starvation is not possible under special accesses. On the same topic, the authors don't mention that their solution prevents livelocks.

Conclusion Their optimistic methods are a very clever idea in the case where worst situations happen only very rarely. I think one can improve their scheme by simplifying the common cases while detecting and handling the worst case situations in an appropriate manner.