VoltDB

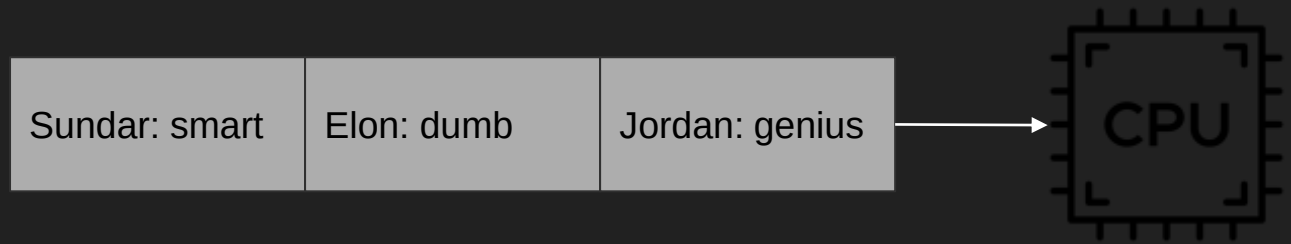
Background

VoltDB is another example of NewSQL database that uses a variety of optimizations in order to try to gain performance increases over traditional solutions such as MySQL or PostgreSQL.

VoltDB design philosophy

Recall: Traditional SQL databases use two phase locking in order to implement transaction serializability, which leads to a variety of issues such as slow throughput due to locking, as well as the need to detect and revert possible deadlocks.

Instead, VoltDB has chosen to use actual serial execution: each database instance actually executes every operation in a single threaded manner, hence no locks are needed!



Supporting Single Threaded Execution

In order to ensure that each database node is capable of running each operation as fast as possible (in order to avoid bottlenecks), VoltDB does the following:

- All data stored in memory
- Transactions must first be stored as stored procedures

Storing Data in Memory

Storing data in memory eliminates disk stalls:

- Less access to disk, no overhead moving pages from disk to page cache
- Command log in conjunction with occasional snapshot ensures durability
  - Command log is similar to write ahead log, but has lower overhead because it just describes the transaction instructions themselves instead of the result (less load on disk)
  - Assuming that command log is updated synchronously, writes can be lost if updated concurrently with memory

Stored Procedures

Using stored procedures eliminates user stalls:

- Occurs in a read modify write transaction when a user has to perform some logic based on the original read, adds a bunch of latency to have to send query results/instructions multiple times over the networks
- Sending a single stored procedure ensures that all of the time spent in the transaction will be because of CPU cycles, not network latency

Partitioning and Replication

- Because data can only be stored in memory, it is more or less inevitable that the dataset will need to be sharded
  - Transactions that reach multiple shards will use one of the nodes as a coordinator node
  - In an ideal world the majority of transactions will be on just one node
  - For some tables that are small but frequently joined on, it makes sense to keep them whole and store them on the same nodes that hold some partitions
- Replication can be done both in a single leader manner or multi-leader manner for cross datacenter operations
  - In the event of multi leader replication, conflicts are resolved by giving deletes precedents over inserts, and if there are two conflicting inserts or updates, using last write wins

Conclusion

By removing locks, VoltDB is able to achieve extremely high performance, even more so than in-memory key value stores (which still may use locking to some extent for lightweight transactions).

While VoltDB itself isn’t overly useful for system design interviews, it is a great case study in how actual just using a single threaded database can be feasible under certain circumstances!