An evaluation to distributed concurrency control

Due to the huge number of transactions, distributed concurrency has achieved a lot of traction. However, multi-partition serializable concurrency control have serious penalties associated with it which can cause scalability issues. It is a significant task to achieve no-shared approach as seen in “Eliminating unscalable communication in transaction processing” paper. This paper provides a thorough study to identify for which workload the distributed concurrency is benign and for workload it can be costly. It quantifies six distributed concurrency protocol unlike the the traditional one or two alternative like the two-phase locking. It uses main-memory DBMS evaluation framework called Deneva which is an open-source and extensible framework. It has multiple client and server in shared nothing fashion on public cloud infrastructure and uses nanomsg to communicate between instances using TCP/IP. Servers are arranged using consistent hashing and clients are aware of partition mapping that are unchanged during execution. It has a custom DBMS engine to avoid the unnecessary overhead. It does not support replication or fault tolerance so only includes scenario which are failure free(isn’t that ironical). It supports extensible modules so new protocols can be added. The reconnaissance step pre compute this information for any protocol if they need it. The evaluations are done on the EC2 framework. When a transaction is aborted, it restarts after the exponential back-off penalty(10 ms). Once the transaction is completed, the coordinator sends an acknowledgment to the client. The lock coordinator in Deneva is CALVIN which is implemented on the CALVIN framework. It was designed to mitigate the effect of two round trip of 2PC. Calvin can perform well when the skew is under certain point, but under high skew(concurrent transaction to access the same record) optimistic concurrency control is the better alternative. The paper focuses on concurrency control over partitioned database and plans to extend beyond the six concurrency protocols. It also aims that the solutions such as RDMA discussed in 5.2 are integrated into Deneva to allow a fair evaluation. It concludes that the distributed transaction on a cluster exceeds the throughput of non-distributed transaction only by a small amount.

Comments:
For a paper to claim evaluating “distributed” concurrency control, isn’t fault tolerance one of the important parameter that is overlooked here?
The scalability according to the results are mainly limited by the commit protocol. Since it only focuses on strong serializability, probably adopting other protocols like Snapshot isolation or Read committed would help improve the results. The authors do mention that isolation level “can” compromise application integrity but I think the paper should have also focused on the protocol that are mostly popular. Also there is a probability that some of the evaluation are skewed because of how Deneva is implemented. Also since the evaluations are on EC2, it is a shared

network where the results can also be affected by the workload by other users. (I remember glancing on a paper that mentioned how experiments performed on the cloud are skewed due to the shared space but I don’t remember its name).