Transactions: From Local Atomicity to Atomicity in the Cloud

(Summary)

Atomicity was developed from "fault tolerant systems" approach. The Newcastle Reliability Project team had invented a concept "Recovery Blocks". It allows a program to return cleanly to an earlier state and execute different block when its execution fails. This is "A" in ACID. Meanwhile, IBM System R team connected the term "transaction" with Atomicity. Databases added Durability to transactions.

Because databases deal with data, scalability has been a big issue. The demand for data has increased. Therefore, databases began to support multi-computer deployments. Databases "scale out" to multiple computer systems instead of "scaling up" with process power. Two approaches are possible. They both need to partition the data. One is Shared Nothing approach. In this approach, databases execute in isolation from other databases. No coordination with another system is required. Therefore, 2PC is needed to coordinate a distributed transaction. However, it introduces extra latency and could block 2PC completion. The other is Data Sharing approach. It is characterized by multiple nodes being able to "share" common data. It is required to cope with distributed cache coherence, distributed locking, and coordinated access.

Although data sharing system was introduced, scalability didn't come to dealing effectively. This is the moment that cloud has been introduced which supports thousands of machines with high speed, relatively low latency communications. Every distributed infrastructure has started by defining standard 2PC. However, 2PC is rarely used since it is blocking protocol and message latencies can be huge. Because of CAP theorem, cloud providers hoped that "eventual consistency" would be enough. However, improving this architecture is ongoing. If we don't do 2PC commit with disks when data is distributed across several disks. we could utilize Virtual Disk concept provided by cloud infrastructure. However, it also has some problem. It is required to all caching be done at a single node.

Deuteronomy has been introduced to overcome some limitations of previous approaches. It divides a database engine into transaction component (TC) and data component (DC) and executes transactional functionality only on TC. It applies distributed and partitioned caching without requiring 2PC by creating TC-DC architecture. In Deuteronomy scaling, it is not required to perform 2PC on disks. Also, transactional scalability can be improved by increasing number of TC's. This will require 2PC commit among the participating TC's.