Eliminating Unscalable communication in transaction processing

Review:

Due to the Moore's law, it is difficult to make clock speed faster for a single core. Hence, the trend has shifted and instead increasing the number of cores in the systems(multicore). In order to utilize the multicore hardware effectively, it is necessary that the software is able to exploit parallelism. In this paper, we focus on the transaction processing workload. Due to the nature of the transactional workload, they provide a lot of opportunity for parallel execution. However, this can get affected due to communication bottleneck. There are three forms of communication: unbounded, fixed and cooperative. The unbounded communication causes issue with the scalability. To ameliorate the effect of unbounded communication, the paper suggest some solution such as locking, logging, physical page accesses and buffer pool frame access. Workloads such as banking are challenging to parallelly execute because they are update intensive(updates must be serialised to maintain correctness), concurrent. The database engine are suppose to maintain the ACID property. ACID properties are the contributor to the complexity. Relaxing consistency reduces amount of communication but is not allowed for large number of system. Shared-nothing architecture eliminates virtually all communication but impacts of shared cache and impose large resource footprints. Eg. VoltDB cannot be truly shared nothing because the ability to broker communication between agents that connect the database. Shared everything architecture requires concurrency management to maintain consistency. The concurrency techniques do not address the communication overhead. Hence data-oriented processing is suggest which combines the properties of shared everything and shared nothing that focuses on eliminating the unbounded communication. Technically, if the transactions are independent, there should be no contention. However, section 2.2 shows the issue faced due to the small bottlenecks and uses Amdahls law to justify it. Logging is serious issue in transaction processing system. Aries itself is serial but can provide high level of concurrent behavior. However, scalability is a bottleneck due to its centralized behavior. Contention in log manager is also an issue here. Logical locking and locking cause severe issue with scaling. Page latching and buffer pool can also cause issue with scaling but effect is comparatively less. Since shared nothing architecture is not feasible completely, we can instead attenuate the unbounded communication by either downgrading it to fixed or cooperative form, avoid it by shared caching, or re-architect by targeting contention prone algorithms. One such technique of re-architect is using DORA which avoids conflicts by using logical partition. This assures that the database locks is managed privately, without distributed transaction and load balancing. Figure 9 shows the benefit of DORA. Shared nothing would achieve similar results however the physical partition has two significant issues: 2pc and non uniform data access. The DORA design is also extended to get the benefits of physical accesses by PLP and overlay buffer pool. Hence, inorder to effectively utilize the modern multicore

hardware, we can reduce the communication patterns which affect scalability to reduce the contention among the system.

Comments:
Shouldn’t re-architect cause significant engineering effort and can lead to unnecessary problems?
Shouldn’t data reshuffling happen when the transactions are further divided ? (Something like the issue with map reduce)