Chiller: Contention-centric Transaction Execution and Data Partitioning for Modern Networks (Technical Perspective)

Alan D. Fekete
University of Sydney
alan.fekete@sydney.edu.au

Many computing researchers and practitioners may be surprised to find a "research highlight" which innovates on the way to process database transactions. Work in the early 1970s, by Turing winner Jim Gray and others, established a standard set of techniques for transaction management. These remain the basis of most commercial and open-source platforms [1], and they are still taught in university database classes. So why is important research still needed in this topic? The technology environment keeps evolving, and new performance characteristics mean that new algorithms and system designs become appropriate. This perspective will summarise the early work, and point to how the field has continued to progress.

The transaction abstraction: Ongoing research into the mechanisms has not changed the essential properties that users and application programmers depend on. We are dealing with data, stored in one (or more) systems. Often these are SQL database management platforms which support rich predicate-based selection commands, but some could be simpler key-value stores. Several data access operations can be grouped together into a *transaction*; this means that the whole group together acts to perform a single change in the real-world domain which is reflected in the data. For example, a real-world action of hiring an employee can be performed through a transaction with several statements: it checks entries in the References table, inserts a new record in an Employee table, and also modifies information in Managers, Department, and so on. The acronym ACID describes the essential properties: the transaction should be *atomic* (all the changes in data from these operations are performed - in this case we say the transaction has committed, or else none of the changes are evident in the data), *consistent* (the changes collectively should not violate any integrity conditions on the data), *isolated* (there should not be interference between concurrently active transactions), and *durable* (changes made by a committed transaction should not be removed unless another transaction explicitly does so). Being consistent is a responsibility of the application programmer, while the others are to be enforced by the data storage systems. There are various levels of isolation, of which serializability is the strongest, where the transactions appear to run one after another.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Copyright 2021 ACM 0001-0782/08/0X00 ...$5.00.

The classical implementation techniques (1970s-1990s): The concept of transaction was supported by several implementation techniques for disk-based stores. The main ideas were used in the prototype relational database management systems such as IBM's System R Practice settled around log records to allow durability despite crashes that corrupt in-memory data, and locks to control interleaving of concurrent activity. For uniform outcome of transactions across multiple stores, the two-phase commit protocol became standard. An elegant theory was also established to reason about all these techniques. The rich theory and practice of transaction management, as they emerged from this period, are described in the reference [2].

On-going innovation (2000 and beyond): Just a few of the important topics in recent years are mentioned here. The increases in capacity of main-memory led to exploration of deterministic concurrency control, where the transaction ordering is not determined dynamically as transactions run and are perhaps impacted by the long delay fetching data from disk. Instead, some fixed rules are used to decide which of the transactions (in a small epoch of time) are run first. The very long message latencies, when data is geo-distributed, have led to attempts to combine the processing of replica consistency, lock management, and commit protocols; an alternative approach to this challenge has been a rise of interest in weak isolation and consistency properties.

Chiller: The following research highlight presents a design called Chiller, driven by new high-bandwidth communication technologies that are becoming common in data centers, which the authors show shift where performance bottlenecks occur in the system. They explore the ways lock holding, replica update, and commit processing, all interact in this new hardware environment. A central insight is to identify which data items are contended, and the system re-orders operations so locks on the contended items are held as briefly as possible. To make this work well, they propose new protocols and also show how this changes the decisions that place data items in different machines for optimal performance.

1. REFERENCES

[1] J. M. Hellerstein, M. Stonebraker, and J. R. Hamilton. Architecture of a database system. *Found. Trends Databases*, 1(2):141–259, 2007.

[2] G. Weikum and G. Vossen. *Transactional Information Systems: Theory, Algorithms, and the Practice of Concurrency Control and Recovery*. Morgan Kaufmann, 2002.