Next Generation Consistency Enforcement

Zechao Shang
The Chinese University of Hong Kong
zcshang@se.cuhk.edu.hk

When more than one worker can access data simultaneously, they usually break data consistency.[1] To enforce data consistency, a huge performance penalty must be paid. This divides system architecture into two directions. One direction considers data consistency less important than "greater goods", such as availability or partition tolerance, and so sacrifices it. Dropping data consistency enforcement components from a system usually leads to significant performance improvements. However, data consistency is still indispensable and ubiquitous in the real world applications whether the data systems support it or not. As a result, researchers and practitioners started to consider the other direction: high performance consistency enforcement. Recent representative works include data-flow systems, transaction processing on partitioned database and invariant-based concurrency control, etc. Unfortunately, although these studies have achieved considerable progress, there is still a giant performance gap between them and the systems without data consistency at all.

Why is protecting data consistency so hard? Based on previous experience, we believe the "no man left behind" promise is the root of all problems. As in most of the RDBMS textbooks, a typical use of databases is online banking. Data consistency (as a part of ACID) is enforced to prevent money being lost. It is a banker's public relation nightmare if even just one customer's account is compromised, so every constraint on every tuple must be respected. Recent protocols build on this same commitment; for example, invariant-based concurrency control protects all invariants that it is capable of.

However, we are not in the 20th century any more. People have a huge volume of data now. The ideal story is to protect perfect consistency on big data. As this is not feasible, everyone must choose between perfect consistency and big data, and maintaining "no man left behind"-style consistency becomes questionable. We envision the next generation system that protects data consistency when consistency is as important as scalability and performance. One potential scenario involves analytical jobs. There are more and more analytical jobs being processed. Parallel and/or distributed analytical algorithms also have the requirement to protecting (intermediate) data integrity from workers' mutual interferences but a slight loophole (caused by inconsistency) is not a fatal mistake. Therefore, next generation consistency enforcement should switch from the paradigm of protecting precious vulnerable data to preventing huge mistakes in connection with big data. In other words, we should aim on provide *average case* consistency enforcement instead of *worst case* consistency enforcement.

There are rich research opportunities regarding next generation consistency enforcement. The first is what *promises* shall the system provide to the end-users. On the one hand, we must pay attention to the performance: the promises are supposed to be easier to enforce than precise consistency levels. On the other hand the end-users, who are generally not system experts, are capable of learning about and manipulating the new promises easily and, more importantly, of understanding how to adopt the new promises to their analytical jobs. Specifically, if the new consistency can not guarantees serializability at all time, how does consistency anomalies affect the accuracy of the analytical tasks. "Preventing 99.99% of cycles in dependency graph" is a bad example of consistency promise: end-users do not know what a dependency graph is or how much damage the remaining 0.01% cycles could cause. A tunable consistency controller that prevents $x\%$ ($0 \leq x \leq 100$) cycles is much better. Understanding two rules is sufficient to utilize this controller: the larger x the slower the system works, and $x = 100$ means serializable. Machine learning developers use a smaller x in earlier iterations for faster computation and $x = 100$ in final iterations for finer convergence in their optimization algorithms.

Efficient implementation is also key to success. In the last example, a relative low overhead when $x = 100$ is important as well as an absolute low overhead when $x = 0$. The implementation must also consider fairness. Empirical studies from some optimization algorithms show the result quality loss is significantly less when the undesired behaviors occur evenly on all data items instead of when they are concentrated on particular items. It is also interesting to investigate the relation between next generation consistency and approximated hardwares, which give up the most basic consistency in favor of better performance, larger capacity and/or low energy consumption. How to effectively utilize them as database storage is challenging. There is also a gap between the state-of-the-art database theory, which is also built upon the "no man left behind" paradigm, and analytical algorithm correctness. Current knowledge of the relationship between weaker consistency and analytical result quality is limited by specific algorithms with special properties such as convexity. We need to extend the theory and expand the known results to connect next-generation consistency with analytics.

[1] We refer to general data consistency, which may include the consistency and isolation in ACID, and single item consistency in replicated databases.

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235