WHITE PAPER

A Brief History of NoSQL

And why it may not be your best choice

The problem NoSQL was created to solve

Some people say Google's BigTable changed everything.

The NoSQL "movement" was a response to the status quo of the data management world in the early 2000s. Web 2.0 companies were struggling to handle the hundreds of thousands to millions of visitors to their websites. Many of these companies did not charge users for access to their websites so they were looking for other ways to make money and saw data as a key to being able to do that. They wanted to collect as much data as possible about their visitors with the hope of being about to achieve some kind of critical mass in that data that could be monetized.

That may not be your problem

The best data management solution for your use case is very dependent on that use case. That seems obvious but we have seen many customers take the approach of buying into using the Google/Facebook/Twitter/LinkedIn "hammer" and then they see every one of their use cases as a nail. That is, they thought all those smart computer science PhDs working at those cool companies surely created the best data management solution available, so they wanted to use it, too.

Need drives design. Those Web 2.0 companies had some very specific and some unique requirements for their data management platforms. They wanted their "database" to be able to accept lots and lots and lots of data coming at them very quickly. But these companies didn't know or really care about what kind of data. There would be some standard and typical data like what you would find in most server logs (IP addresses, browser type, cookie id, etc.) which would all fit quite nicely into a relational database schema but they also wanted to be able to capture unstructured data and have the flexibility to add new capture items in the future.

Back to Google — they had a very specific need for being able to collect link information from their web crawlers to help generate the foundation for their search rankings. They weren't trying to create a highly reliable database to track money or to do billing for mobile phone calls or allow real-time A/B testing for online games.

Why VoltDB's NewSQL approach may be better

Michael Stonebraker and his team had designed many of the world's leading databases. They had just solved a major challenge for analytical database with Vertica — organizing data records on disk based on their columns rather than their rows that allowed analytical queries to perform much better than the traditional rowstore. That was because they knew that most analytical queries did disk scans to retrieved data from lots of rows but from only one or two columns. By changing the default storage format, they eliminated a major bottleneck in retrieving data from disk.

This team then turned their attention to transactional systems to try to identify and then eliminate the bottlenecks that were the limiting factors on OLTP database performance. Disk-based OLTP was already about as fast as it could get. The data access pattern for disk-based OLTP databases is different than that of disk-based OLAP databases. Instead of doing massive scans to retrieve data from a limited number of columns, OLTP systems tend to go to individual records for CRUD operations (Create, Read, Update, Delete), so row-stores are very efficient layouts for OLTP databases. At that time, DRAM prices were starting to be more affordable so Stonebraker and his team looked at how to make an optimized transactional database that operated primarily in-memory.

When they examined the popular OLTP databases of the time, they found that contention control and management was the main factor limiting the speed of those databases; because the database needed to keep simultaneous users from corrupting the database, things like buffer pool management, locking, latching, and logging (for crash recovery) were needed. If you could eliminate those things, an OLTP system could go much faster.

VoltDB was designed to eliminate those things. Each VoltDB database instance is optimized for a specific application by partitioning the database tables and the stored procedures that access those tables across multiple "sites" or partitions on one or more host machines to create the distributed database. Because both the data and the work is partitioned, multiple queries can be run in parallel. At the same time, because each site operates independently, each transaction can run to completion without the overhead of locking individual records that consumes much of the processing time of traditional databases. Finally, VoltDB balances the requirements of maximum performance with the flexibility to accommodate less intense but equally important queries that cross partitions. By using serialized processing in each partition, VoltDB ensures transactional consistency without the overhead of locking, latching, and transaction logs, while partitioning lets the database handle multiple requests at a time.

Because VoltDB did not throw out the basics of relational databases in pursuit of speed and scalability but rather improved on them, it retains the benefits of relational systems such as strong ACID compliance and immediate consistency unlike popular NoSQL platforms.

Today, modern, data-driven applications need to work at high speed and to scale for high volume. While NoSQL solutions can operate very quickly and scale to handle high volume, they compromise on key factors to achieve that speed and scalability — they cannot provide consistent operations like transactions at scale like VoltDB can.

Applications using VoltDB that require scalable transactions include:

- Allowing a mobile call to connect while verifying the user's balance
- Placing a trade at the best offer price
- Presenting a mobile ad to thousands of users while strictly adhering to the customer's ad budget
- Managing tight SLAs for telco providers
- Detecting a fraudulent card swipe before the transaction is approved

NoSQL databases are generally not the right choice for these types of applications.

Conclusion

Both NoSQL and NewSQL technologies like VoltDB provide datastores on which highly scalable applications can be built. NoSQL datastores are a great choice for applications where availability — getting a response — is valued more highly than getting a consistent, correct response at all times. NewSQL systems provide applications with scalability as well, and also offer strong consistency and transactional interaction, favoring consistency over availability in failure scenarios.

While nearly all NoSQL solutions deliver scalability, VoltDB delivers on scalability and adds strongly consistent transactions; further, few can NoSQL solutions match VoltDB's combination of blazing speed, transactional consistency and scalability.

VoltDB, the most mature and proven NewSQL system, is a cloud-ready SQL operational database. VoltDB combines real-time analytics on inbound data feeds with strong ACID transactions, native clustering, and Hadoop ecosystem support. This allows VoltDB to be the "single source of the truth" for today's data-intensive applications. It's a great choice for use cases where very high performance and predictably low latency are critical as well as where accurate counting/accounting is important, such as in policy enforcement, personalization, fraud/anomaly detection, and other request-response style fast-decisioning and fast data pipeline applications.

VOLTDB