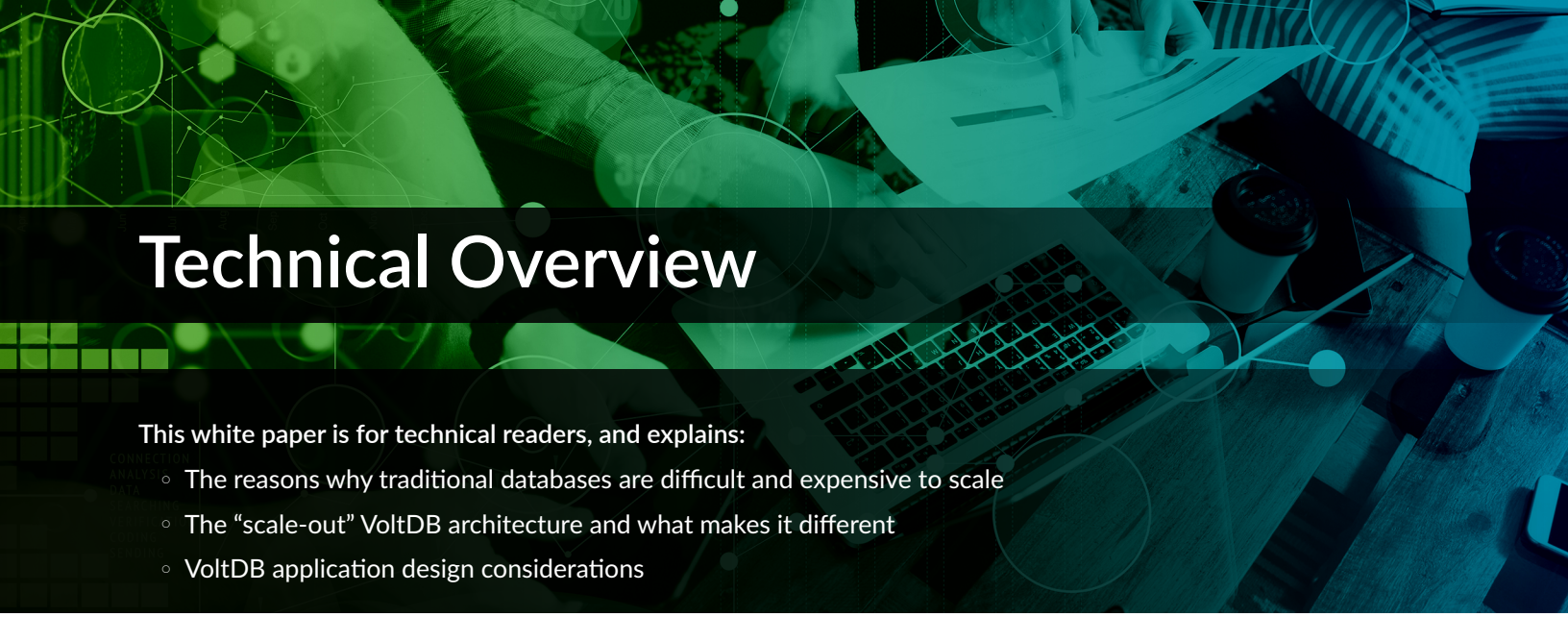


Technical Overview

This white paper is for technical readers, and explains:

- The reasons why traditional databases are difficult and expensive to scale
- The "scale-out" VoltDB architecture and what makes it different
- VoltDB application design considerations

High Performance RDBMS for Fast Data Applications Requiring Smart Streaming with Transactions

Overview

Traditional databases (DBMS) have been around since the 70ies, they offer proven standard SQL, ad hoc query and reporting, along with a rich ecosystem of standards-based tooling. However, these legacy DBMSs are hard to scale and offer a rigid general purpose architecture that is not suitable for modern apps.

The NoSQL category of tools was born from the need to solve the scaling problems associated with legacy SQL. Along with scale, NoSQL also offers greater availability and flexibility. However, NoSQL sacrifices: data consistency, transactional (ACID) guarantees, and ad-hoc querying capabilities, while significantly increasing app complexity.

VoltDB is a leader in the NewSQL category of databases that were engineered to offer the best of both worlds. VoltDB combines ACID transactions, SQL capabilities, HA clusters and DR of traditional DBMSs with linear scale-out, virtualization and cloud nativeness of NoSQL. Moreover it allows for capabilities that weren't present in either, such as in-database Machine Learning driven smart decisions on an large event stream for Smart Streaming. This allows VoltDB to be the system-of-record for data-intensive applications, while offering an integrated high-throughput, low-latency stream ingestion engine for streaming workloads. This makes VoltDB the perfect choice for modern apps On-Prem and Public Cloud that require:

- Orders of magnitude better performance than a traditional RDBMS — 100x faster
- Strict, industry best ACID transaction guarantees to ensure data consistency, integrity and accuracy
- SQL as the native data language, simplifying development and enabling richer interaction
- Built-in high availability, fault tolerance and cross-datacenter replication for disaster recovery in geographically-distributed clusters
- Operationalizing Machine Learning based rules on incoming event driven smart streaming
- Cloud native dockerized application deployment through kubernetes integration

For fast, data-driven applications, VoltDB offers substantial performance and cost advantages as illustrated by the results of the TPC-C-like benchmark below, in which VoltDB and a well-known OLTP DBMS were compared running the same test on identical hardware (Dell R610, 2x 2.66Ghz Quad-Core Xeon 5550 with 12x 4GB (48GB) DDR3-1333 Registered ECC DIMMs, 3x72GB 15K RPM 2.5in Enterprise SAS 6GBPS Drives):

	Nodes	VoltDB	DBMSx	VoltDB Advanatage
"TPC-C-like" workload (VoltDB lab)	1	53,000 TPS	1,555 TPS	45x better throughout
	12	560,000 TPS	NA	Near-linear (.9) scal-

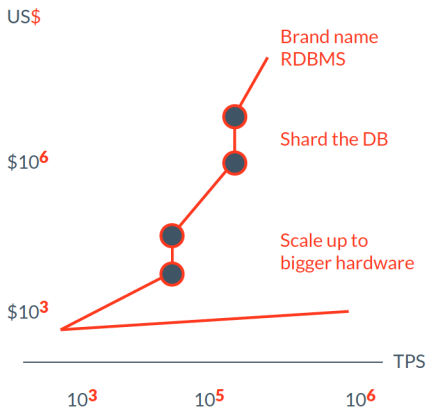
As shown above, VoltDB's massively parallel architecture also provides elastic scalability running on low-cost commodity servers. It allows application developers to scale applications simply by adding servers to a VoltDB cluster, instead of building complex, costly sharding layers. VoltDB's ACID compliance ensures developers never need to sacrifice data consistency to gain high performance or scalability.

RDBMS Scaling Alternatives and Costs

As an application's popularity and usage increases, scaling it up to support heavier operational workloads and run 24x7x365 can be painful. The scalability bottleneck is often the DBMS. To adapt to heavier workloads, an application may experience a number of disruptive scaling events during its lifecycle, including:

- Migration from an inexpensive commodity server to a very expensive server and SAN implementation
- Migration from an open source DBMS to a higher-scale commercial DBMS such as Oracle
- Re-designing the database (and corresponding application data access logic)
- Implementing a data partitioning or "sharding" scheme — manually dividing the database into many smaller databases running on different servers and modifying application code to coordinate data access across the partitions
- Or fedup with Oracle licensing costs, implementing an open source NoSQL/ Key-Value Database (KV store), thereby forfeiting transactional consistency, adding latency and giving up the ability to use SQL

RDBMS Scaling Alternatives and Costs



Dealing with a fast-growing user workload is a good problem to have, but the popular "scale-up" approaches to scalability are costly, requiring expensive hardware and DBMS upgrades. Scale-up approaches also add development complexity, and increase overhead and maintenance costs. And no matter what scheme is used, the opportunity cost is also high — time spent fixing performance problems means less time spent implementing higher-value business functionality.

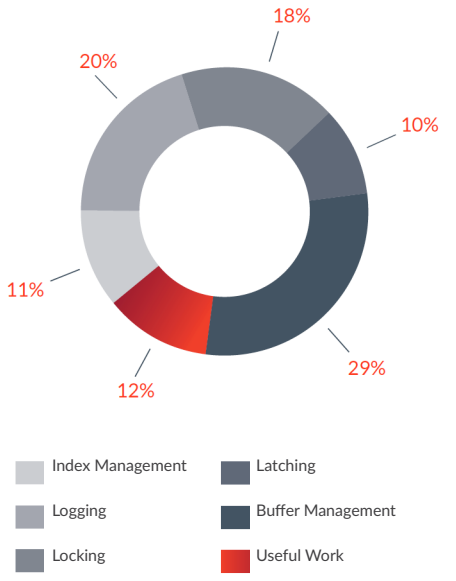
What's needed is a DBMS that "scales out" linearly and limitlessly by adding new commodity servers to a shared-nothing DBMS cluster. However the "scale out" must not come at the cost of compromising transactional consistency, giving up on SQL or adding latency to the workload which is the case with NoSQL databases.

Let us look at why neither, traditional, not NoSQL DBMSs are the right for the job.

Why Do Traditional DBMSs Have Difficulty Scaling?

In 2008, a team of researchers led by Mike Stonebraker published a seminal paper in the ACM SIGMOD entitled "OLTP Through the Looking Glass, and What We Found There." The paper exposed key sources of processing overhead that plague traditional disk-based RDBMS products, and concluded that removing those overheads and running the database in main memory would yield orders of magnitude improvements in database performance. Along with other noteworthy research of that period, Through the Looking Glass signaled a sea change in thinking around in-memory operational systems. Traditional DBMSs have five sources of processing overhead:

Gereral Purpose RDBMS Processing Profile



- **Index Management:** B-tree, hash and other indexing schemes require significant CPU and I/O.
- **Write-ahead Logging:** Traditional databases write everything twice; once to the database and once to the log. Moreover, the log must be forced to disk to guarantee transaction durability. Logging is, therefore, an expensive operation.
- **Locking:** Before touching a record, a transaction must set a lock on it in the lock table. This is an overhead-intensive operation.
- **Latching:** Updates to shared data structures (B-trees, the lock table, resource tables, etc.) must be done carefully in multi-threaded environment. Typically, this is done with short-duration latches, which are another considerable source of overhead.
- **Buffer Management:** Data in traditional systems is stored on fixed-size disk pages. A buffer pool manages which set of disk pages is cached in memory at any given time. Moreover, records must be located on pages and the field boundaries identified. Again, these operations are overhead- intensive.

Designed originally for data integrity, this overhead prevents traditional databases from scaling to meet contemporary data volumes and workloads.

Why Aren't NoSQL DBMSs a Good Compromise?

A band-aid solution employed today is to build an Open Source technology stack by layering a message bus like Kafka along with a NoSQL database. At first glance this stack may seem like an inexpensive fix as it allows linear "scale out" rapid application development through key-value semantics and inexpensive scale on virtual infrastructure. However distributing data across lots of virtual DBMS instances require coordination for ACID compliance across the data spanning these instances. Not to mention, none of the NoSQL implementations have a decent SQL implementation forcing applications to be completely re-written. Not having a DDL and a schema makes scaling further challenging. Over the last year or so, NoSQL DBMS vendors have added some rudimentary multi-entity transactionality and t ery language. The reality is that none of these solutions work for mission critical applications at larger scale. They leads to a host of new issues, such as: 10x worse latency accumulated over multiple layers, application complexity, and rampant hardware sprawl with 100s of nodes per cluster.

VoltDB Eliminates Traditional DBMS Overheads While Preserving ACID Unlike NoSQL

With VoltDB:

- Data and the processing associated with it are partitioned together and distributed across the CPU cores ("virtual nodes") in a shared-nothing hardware cluster
- Data is held in memory for maximum throughput and to eliminate the need for buffer management
- Each single-threaded partition operates autonomously, eliminating the need for locking and latching
- Data is automatically replicated for intra- and inter-cluster high availability and disaster recovery
- Analytics is done close to the data for complex low latency real-time decisions.

VoltDB Architecture

VoltDB leverages the following architectural elements to achieve its performance, scaling and high availability objectives:

- Automatic partitioning (sharding) across a shared-nothing server cluster
- Main-memory data architecture
- Elimination of multi-threading and locking overhead
- Clustering for High Availability and Active Active Replication for Disaster Recovery
- Ad-hoc SQL access via JDBC and client language bindings
- Multi-statement transactions via Stored Procedures
- In-database Machine Learning driven Smart Streaming
- Cloud native container based application deployment with Kubernetes
- Native exports to OLAP, Data Warehousing and Hadoop Data Lakes

Automatic Partitioning Across Shared-nothing Server Clusters

The size of databases is increasing at the rate that business expands. For example, a purchase order database increases in size at the rate that active purchase orders increase. In almost all database applications, this rate is slower than the rate at which main memory decreases in price. Therefore almost all database applications are (or will soon be) candidates for main-memory deployment.

VoltDB is a main memory DBMS.

In VoltDB, each partition is stored in main memory and processed by its associated, single-threaded execution engine at in-memory speed. In-memory processing eliminates disk waits from within VoltDB transactions, along with the need for buffer management overhead.

VoltDB can save data snapshots and command logs to disk for backup and recovery purposes and also export data to a data warehouse database for analysis and querying. VoltDB implements a concept called command logging for transaction-level durability. Unlike traditional write-ahead logs, VoltDB logs the instantiation of commands to the database rather than all resulting actions. This style of logging greatly reduces the load on the disk system while providing either synchronous or asynchronous logging for transaction-level durability.

Elimination of Multi-threading and Locking Overhead

Conventional databases experience disk and user stalls within transactions. Rather than letting the CPU idle during stalls, those DBMSs interleave SQL execution from multiple transactions during waits so the CPU is always busy. This requires much complex latching and locking overhead.

VoltDB doesn't experience user stalls (since transactions happen within stored procedures) or disk stalls (because VoltDB processes data in main memory). Therefore, VoltDB is able to eliminate the overhead associated with multi-threading (latching) and locking. Each VoltDB execution engine is single-threaded and contains a queue of transaction requests, which it executes sequentially—and exclusively—against its data. Elimination of stalls and associated locking and latching overhead allows typical VoltDB SQL operations to complete in microseconds.

For single-partition transactions, each VoltDB engine operates autonomously. For multi-partition transactions, one engine distributes and coordinates work plans for the other engines. VoltDB assumes that an application designer can construct a partitioning or cloning scheme and transaction design that localizes a large majority of transactions to a single virtual node. Many fast data applications such as telco billing, personalization, game state, sensor management, and capital market risk share this profile.

Clustering for High Availability and Active Active Replication for Disaster Recovery

VVoltDB achieves high availability for 24x7x365 operations very simply and economically through automatic intra-cluster and inter-cluster replication. Data is synchronously committed to replicated partitions within the cluster before transactions commit. This provides durability against single-node failures. To provide durability against cluster failures (e.g., in the event of a data center catastrophe), transactions are asynchronously committed to a replica cluster (usually in a different geographic location).

VoltDB automatically guarantees that every replica, whether in the same cluster or a different one, runs transactions in the same order and thereby achieves the same global order.

VoltDB runs an active-active configuration within a cluster. If a node failure occurs, VoltDB automatically (and seamlessly) switches over to a replica node. Hence, an application is unaware that a problem occurred. Recovery is performed automatically; the recovered node (or cluster) queries the running VoltDB cluster to recover its data.

VoltDB's passive database replication feature, parallel binary replication between the master and replica clusters, has been augmented with geo-distributed, cross-datacenter replication (XDCR). This bidirectional database replication, also called active-active replication, allows enterprises to maintain separate, active copies of the database in two separate locations.

Ad-hoc SQL Access via JDBC and Client Language Bindings

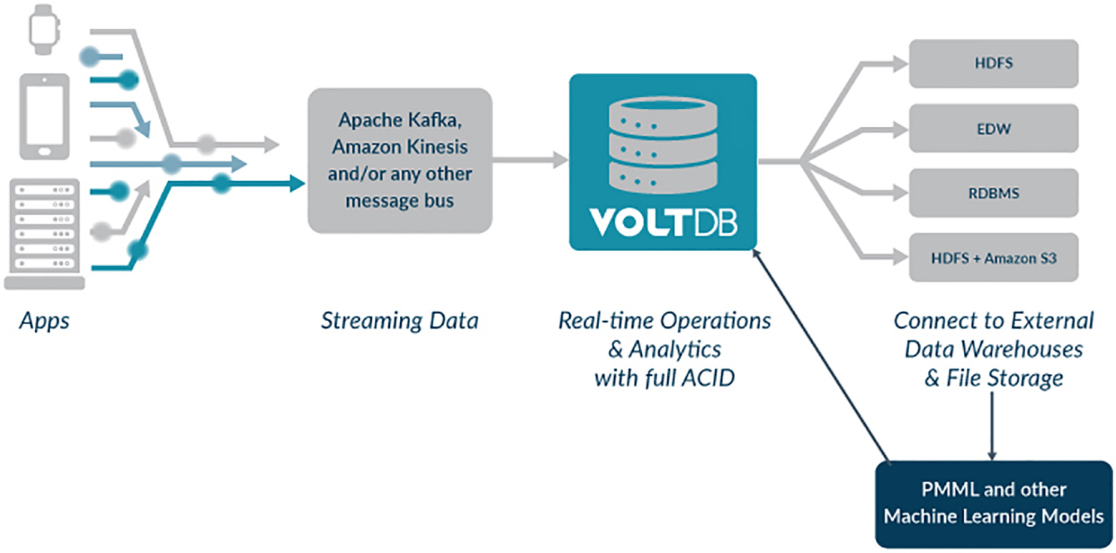
VoltDB exposes stored procedure, ad hoc and JDBC interfaces. With stored procedures, only a single round trip occurs between the client and the server per transaction. Hence, the run-time interface to VoltDB is to execute a stored procedure, substituting transaction-specific constants for parameters.

Stored procedures are written in Java, with embedded SQL calls for database services. VoltDB supports ANSI-standard SQL with application-enhancing extensions.

In-database Machine Learning Driven Smart Streaming

Legacy database technology was focused on analyzing historical data to gain an rear-view understanding of business performance. While it is important to analyze where the business is coming from, in order to gain the competitive advantage and differentiate your application it is critical to utilize deep learning and take action in-event; with the end goal of driving desirable business outcomes.

Building a robust predictive analytics model is only half the battle. Utilizing the model in production for real-time decision making on streaming data is a key element of Machine Learning, that is often overlooked because of the limitations of legacy technology. VoltDB's VoltML module allows you to operationalize ML models in real-time. With VoltML, you can deploy complex Predictive Model Markup Language (PMML) models in production easily and quickly without having to write any code. VoltML utilizes simple SQL queries to take real-time intelligent decisions on streaming data. PMML models can be deployed seamlessly in just minutes, allowing for: reduced analytical model processing times, frequent model updates, ensuring optimal accuracy and reliable decisions.



Cloud Native Container Based Application Deployment with Kubernetes

Kubernetes was designed for stateless web apps that can easily spin up new interchangeable instances in case of node failure or scale-out. Up until recently, orchestrating SQL databases in Kubernetes had been challenging as operational database systems store state and can't just be spun up or spun down on a moment's notice.

VoltDB however, was architected to run in an distributed containerized environment. Enabling VoltDB to be one of the first operational databases to offer support for Kubernetes. The VoltDB Kubernetes module allows organizations to automate the deployment of VoltDB clusters, with the Kubernetes module DevOps teams can seamlessly spin up and spin down VoltDB docker clusters, monitor running clusters and update clusters on an ongoing basis.

Kubernetes orchestration turns many tedious and complex tasks into something as simple and declarative as a config file, allowing for continuous and frequent deployment. The VoltDB's Kubernetes module fits right into your automated CI/CD pipelines, accelerating the development and deployment of apps that rely on a fast operational back end database.

Native Exports to OLAP, Data Warehousing and Hadoop Data Lakes

VoltDB is specially designed to handle fast, large volumes of operational database transactions. As an in-memory database, VoltDB is not well suited to the petabyte + data volumes increasingly found in data warehousing. However, organizations often want to store and process the information collected by VoltDB in an OLAP system, data warehouse or a Hadoop based data lake. Therefore, VoltDB includes an export integration subsystem that spools VoltDB data to such system to enable deep reporting and analysis via these tools. It also allows for using this downstream system to train the machine learning models that can be imported into VoltDB for in-event smart streaming.

VoltDB Versus Other DBMS Alternatives

VoltDB isn't the first attempt at overcoming the performance and scalability limitations of traditional databases. Two alternatives to VoltDB include running conventional databases in memory, or using a "NoSQL" key-value store (KV store).

Alternative in-memory systems may remove buffer management and logging (at the expense of durability). Even with these features removed, the maximum performance improvement is roughly 2x. To achieve the 50-100x speedup of VoltDB, all legacy OLTP time-syncs must be removed (buffer management, logging, latching and locking).

To deliver better performance on scale-out hardware, some databases, such as NoSQL KV stores, eliminate some of this overhead — and sacrifice SQL interactivity and data integrity as well (deliver- ing "eventual consistency"). Unfortunately, since KV stores don't execute SQL (although some NoSQL offerings have begun to provide limited, proprietary SQL alternatives), functionality that would normally be executed by the database must be implemented in the application layer.

Summary

VoltDB leverages observations derived from research and industry expertise about OLTP workloads to achieve linear scaling as nodes are added to a VoltDB cluster. This partitioning, across multiple machines and multiple cores, is central to the design of VoltDB; it mitigates the contention for resources that limit legacy DBMS scalability and addresses the gaps of NoSQL DBMS. The main-memory storage of all data greatly reduces network and disk latency. In-database machine learning enables the operationalization of complex ML models in real-time to take actionable business decisions. While, Kubernetes support allows DevOps teams to automate the deployment of VoltDB clusters. Combined, these departures from the status quo allow VoltDB to offer next-generation NewSQL DBMS scalability, performance, decisioning, and manageability, liberating organizations from expensive shared-memory, shared-disk database systems that do not scale.

Next Steps

To learn more about VoltDB, visit www.voltdb.com where you'll find product documentation, customer use cases, a self help VoltDB University and downloadable, free-trial versions of the technology.

About VoltDB

VoltDB is the only in-memory transactional database for modern applications that require an unprecedented combination of data scale, volume, and accuracy. Unlike other databases, including OLTP, Big Data, and NoSQL, that force users to compromise, only VoltDB supports all three modern application data requirements: **1. Millions** — VoltDB processes a relentless volume of data points from users and data sources. **2. Milliseconds** — VoltDB ingests, analyzes, and acts on data in less than the blink of an eye. **3. 100%** — Data managed by VoltDB is always accurate, all the time, for all decisions. Telcos, Financial services, Ad Tech, Gaming, and other companies use VoltDB to modernize their applications. VoltDB is preparing energy, industrial, telco and other companies to meet the challenges of the IoT. VoltDB was founded by a team of world-class database experts, including Dr. Michael Stonebraker, winner of the coveted ACM Turing award.

20 February2019

VOLTDB