VoltDB for Telco Technical Overview

The operational complexity of many databases, from legacy RDBMSs to open source options, can be daunting. Full-time DBA support isn't an option for many small-medium companies, and can represent a significant seven-figure sum for larger ones. Architectural complexity, scale out vs. scale up issues, HA and cross-datacenter replication, data consistency, cloud-readiness, capacity for virtualization, even old-school locking and latching present issues more familiar to a distributed systems expert than to an app developer or DBA. More importantly, the operational complexity will inevitably bubble up to affect end users.

Many NoSQL offerings, which offer a more flexible approach to scale out, flexible schema and data types, fail on support for scalable transaction support when working with shared, finite resources: credit balances or trade verification, authentication and authorization in telco and finserv, precise billing in ad tech and telco, and in-game personalization in online gaming, to name a few use cases.

Telcos and CSPs build value on operational applications:

- Operations Support Systems (OSS) that support management functions such as network inventory, service provisioning, and network configuration and fault management;
- Business Support Systems (BSS) that provide applications to support customer-facing activities such as billing, order management, CRM, and call center automation; and,
- Real-time applications that enable providers to act instantly to create new services and applications, improve quality of service, meet SLAs, and fulfill customer expectations.

Telco software solutions are frequently in a service provider's call path. Software providers require latencies for physical transactions to be sub 20ms, so performance and scalability are major requirements. VoltDB is the best solution available for ingesting, analyzing and acting on the massive volumes of real-time data streaming from current and future BSS and OSS systems. It combines accuracy, scalability and manageable TCO, even for cutting edge scenarios such as managing network usage, charging, billing and quality of service for many millions of users based in multiple data centers simultaneously.

VoltDB Basics

VoltDB is an in-memory, SQL, cloud-ready operational database for modern applications that require the ability to manage data at unprecedented scale and volume, with 100% accuracy. VoltDB rapidly imports, operates on, and then exports vast amounts of data at lightning speed. Its robust architecture combines the best of traditional transactional databases with the speed and scalability of newer entrants.

Unlike OLTP, Big Data, and NoSQL offerings that force users to compromise, only VoltDB supports all three modern telco application data requirements:

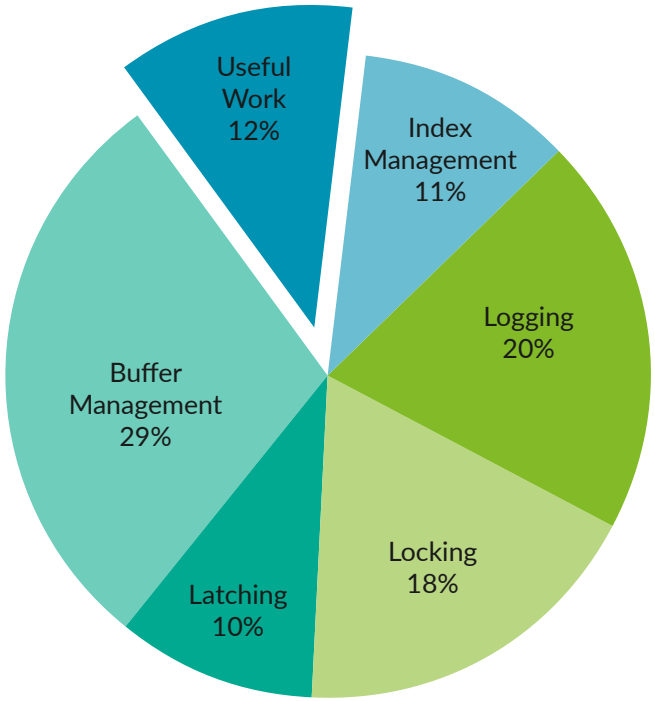
1. **Millions** — VoltDB processes relentless volumes of data from users, devices and sources.
2. **Milliseconds** — VoltDB ingests, analyzes, and acts on data in milliseconds, with predictable low latency.
3. **100%** — Data managed by VoltDB is always accurate, all the time, for all decisions.

Why VoltDB?

Traditional RDBMSs are too slow to handle real-time, request-response, operational applications requiring reliable, millisecond latency responses; lack the ability to scale out; and may have unpredictable licensing costs. NoSQL systems offer linear scalability, flexible schema or schema-less approaches, and in some cases can handle vast streams of incoming data. However, any products or open source NoSQL projects are not SQL compliant, are not operational in nature, are not ACID compliant, and thus are more suitable for analytics workloads.

Technical details

VoltDB was designed by Dr. Michael Stonebraker to address the shortcomings of traditional online transaction processing (OLTP) systems. With VoltDB, Stonebraker and his team were able to eliminate performance issues such as latching and locking, buffer management, and transaction management. For a more detailed look at the decisions behind VoltDB's architecture, read the Technical Overview here.



VoltDB's Architecture

VoltDB was built to bring speed, scalability and performance to operational applications. Below are some of the features that enable VoltDB to meet the needs of modern applications.

Partitioned by core, with a single thread per core

VoltDB partitions workload by CPU core, forcing all work for a given partition towards a single thread running on a single core.

Transactions cannot span invocations

Applications cannot lock records in VoltDB. All transactions begin, do their work and end without ever leaving the core. This means that applications that work with shared finite resources can't trip each other up as they all try to decrement the same counter at the same time. It also cuts out CPU overhead that otherwise would be spent on latching and locking.

Stored Procedures

Because transactions cannot span invocations, stored procedures are required to provide ACID for transactions. Stored Procedures drastically reduce the number of network trips for complicated transactions. VoltDB's Stored Procedures are written in Java, which avoids developers having to learn a new language. JDBC access also is supported.

An open and asynchronous API

VoltDB's underlying client API is asynchronous. It's both published and relatively simple. This works well in a telco context, as it means clients don't have to create more and more threads and connections as workloads go up.

Workload replicated by partition for High Availability

Because VoltDB partitions work by CPU core, HA can be implemented by having two or more different cores to do the same queue of work in the same order. Unlike a legacy RDBMS, where an outage leads to an IO storm as the surviving database nodes try and figure out the deceased node was doing when it died, VoltDB will wait 1-3 seconds to make sure the deceased node is in fact gone and then continue doing the same work on the surviving partitions. This work is hidden behind the scenes from the client.

In-memory only

The only time VoltDB directly reads from disk is when starting. You can, if you really want, configure it to flush to after each transaction, but most customers are happy to use a HA cluster and rely on the fact that transactions will be micro-batched to disk on two separate servers within a few 10s of milliseconds.

"Shared Nothing" architecture, no hardware dependencies

VoltDB does not require a SAN, SSD or shared disk storage to work. It's agnostic in its choice of Linux and supports virtualization. It runs well on private, hybrid and public cloud implementations.

Rule-based optimizer

Query plans in VoltDB never change. This is a significant advantage as the behavior of a SQL statement will not change once deployed.

Feeds to downstream systems

A classic weakness of OLTP databases is the need to issue "SELECT *" queries to unload data for downstream systems to use. VoltDB has an 'at least once' queue mechanism that looks to developers like a SQL table — when you insert into it you create an entry in the queue, which can then be sent to HTTP, Kafka, HDFS, wherever.

'Active-Active-Active' Cross-Datacenter replication

Not only does VoltDB support Active-Active clusters at different locations, as of v7.0 it supports Active-Active-Active clusters. This means that your data can be in multiple locations and the data center nearest the user can handle the request. Obviously, there is a need to manage conflicts in this scenario.

SQL and other methods of communication

- VoltDB supports JDBC-type access, although it is not recommended for production purposes.
- VoltDB's design uses stored procedure calls where you send in a set of parameters and get back a list of results as a transaction.

- Procedure calls are both asynchronous and sent to all the nodes handling the data in question. Each node then executes the procedure in a deterministic manner. Node failures do not require reading from disk to recover, as the surviving nodes are already caught up.
- The asynchronous nature of the procedure call has big advantages in the telco space — at no point does code have to wait for a response while milliseconds pass. An example would be ending a telco session, which can be a 'fire and forget' operation in VoltDB — the application creates an asynchronous call to end the session and continues on its way without having to wait for a confirmation.
- The format VoltDB uses for messages is published, so writing new clients in new languages is easy.
- VoltDB supports C#, C++, Erlang, Go, Java, JDBC, JSON + HTTP (REST), Node.js, PHP, Python, Ruby

ACID compliant

- VoltDB supports 'repeatable reads' - none of the data an application is working with can be changed by anyone else during a transaction.
- Procedure calls are effectively transactions — everything in a procedure either happens or it doesn't.
- VoltDB's HA architecture involves executing the same code with the same inputs on multiple servers at the same time. Even if a node dies in the middle of a transaction, the surviving nodes will complete the same transaction.

VoltDB Disk Interaction/Export Tables

Export tables, which look like tables from a SQL level, are 'at least once' queues to CSV, HDFS, Kafka etc. Use VoltDB adaptors or write your own.