VoltDB The NewSQL database for high velocity applications

VoltDB Technical Overview

A high performance, scalable RDBMS for Big Data, high velocity OLTP and realtime analytics

VoltDB, Inc.
www.voltdb.com
Follow VoltDB on Twitter @voltdb

Table of Contents

Overview

Created by database pioneer Michael Stonebraker along with senior computer scientists from MIT, Yale, Brandeis and Brown universities, VoltDB is a blazingly fast relational database management system (RDBMS) designed to run on modern scale-out computing infrastructures.

VoltDB is aimed at a new generation of high velocity database applications that require:

- Database throughput reaching millions of operations per second
- On demand scaling
- High availability, fault tolerance and database durability
- Realtime data analytics

The table below summarizes a few high-velocity application types. Some applications – financial trade and telco CDR, for example – have been in production for decades. Others reflect markets that are emerging from technology innovations and Web economies. New or old, these systems must ingest high velocity data inputs, provide stateful data management, and support a range of realtime analytics on that data. VoltDB is designed specifically to support these and many other similar applications.

	Data Source	High-frequency operations	Lower-frequency operations
Financial trade monitoring	Capital markets	Write/index all trades, store tick data	Show consolidated risk across traders
Telco call data record management	Call initiation request	Real-time authorization	Fraud detection/analysis
Website analytics, fraud detection	Inbound HTTP requests	Visitor logging, analysis, alerting	Traffic pattern analytics
Online gaming micro transactions	Online game	Rank scores: •Defined intervals •Player "bests"	Leaderboard lookups
Digital ad exchange services	Real-time ad trading systems	Match form factor, placement criteria, bid/ask	Report ad performance from exhaust stream
Wireless location-based services	Mobile device location sensor	Location updates, QoS, transactions	Analytics on transactions

Unlike legacy RDBMS products and NoSQL data stores, VoltDB enables high-velocity applications without requiring complex and costly sharding layers or compromising transactional data integrity (ACID) to gain performance and scale:

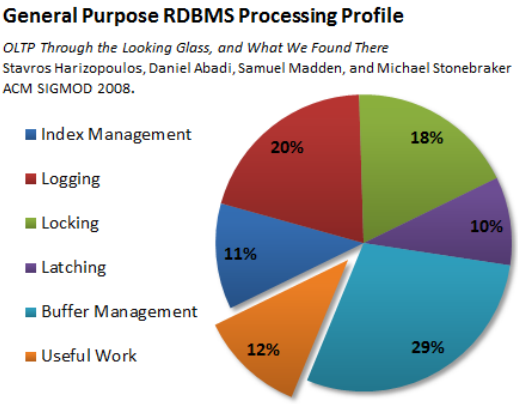
	VoltDB	NoSQL	Traditional RDBMS
Scale-out architecture	✓	✓	
Built-in high availability	✓	✓	
Multi-master replication	✓	✓	
Built-in durability	✓		✓
ACID compliant	✓		✓
SQL data language	✓		✓
Cross-partition joins	Automatic	In app code	In app code
Cost at scale	$	$$$	$$$$

This white paper is for technical readers. It explains:

- The reasons why traditional databases are difficult and expensive to scale
- VoltDB's high throughput scale-out architecture and what makes it different
- VoltDB database design considerations and patterns

The paper also provides information to help product evaluators get started with VoltDB.

Why Traditional Databases Have Difficulty Scaling



Most traditional RDBMS products are based on decades-old designs. They were conceived when OLTP databases supported hundreds of concurrent data entry clerks using an application during the normal work day. Today, databases are expected to support hundreds of thousands of users accessing applications through the Web – 24x7x365. In addition, an increasing percentage of data originates from automated sources such as sensors, scanners and software systems. Conventional databases simply can't scale to meet the throughput requirements of "fire hose" data sources because they were designed to run on older computing infrastructures and to handle modest transaction workloads. Thus, legacy RDBMS products are plagued by overheads that can only be overcome by innovative designs and modern computing platforms. The primary sources of overhead found in legacy RDBMS products are summarized below.

Logging: Traditional databases write data twice – once to the database and once to a data ("write ahead") log for durability. Moreover, the log must be flushed 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 to ensure that no conflicting operations can be performed on that record. Locking is an overhead-intensive operation.

Latching: Updates to shared data structures (B-trees, the lock table, resource tables, etc.) must be done carefully in a multi-threaded environment. Typically, latching is done with short-term duration latches, which are another considerable consumer of CPU resources.

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. Records must be located on pages and the field boundaries identified. Again, these operations are overhead intensive. Designed originally for data integrity, buffer management prevents traditional databases from scaling to handle modern workloads.

VoltDB eliminates the overheads that plague traditional RDBMS products:

- Data and associated processing are partitioned together and distributed across the CPU cores ("partitions") of a shared-nothing hardware cluster.
- Data is held in main memory, eliminating the need for buffer management.
- Transactions execute sequentially in memory, eliminating the need for locking and latching.
- Synchronous multi-master replication provides built-in high availability.
- Command logging provides a high performance replacement for "write-ahead" data logging.

These innovations enable VoltDB to run magnitudes faster than traditional RDBMS products and to scale linearly on low-cost clusters of commodity servers, all while maintaining ACID compliance.

The VoltDB Architecture

VoltDB is designed to optimize the VLSI designs of multi-core CPUs. It exploits clustered server topologies and leverages increasingly abundant main memory to handle high velocity database workloads. VoltDB is a fully ACID-compliant transactional database, relieving developers from having to write custom code to process transactions and manage rollbacks (i.e., database "heavy lifting").

VoltDB leverages the following architectural elements to achieve its performance and scalability gains:

- In-memory storage to maximize throughput, avoiding costly disk accesses
- Serializing all data access, thus avoiding overheads of traditional databases such as locking, latching and buffer management
- Performance and scale through horizontal partitioning
- High availability through synchronous, multi-master replication (in VoltDB parlance, "K-safety")
- Durability through an innovative combination of database snapshots and command logs that store recoverable state information on persistent devices (i.e., spinning disks and/or SSDs)

The remainder of the paper describes how VoltDB works by explaining each of the following topics.

Section	Topics
High-throughput, low-latency SQL database operations	In-memory operation, stored procedure interface, asynchronous/synchronous stored procedure execution and serial transaction processing
VoltDB scaling architecture	Partitioning, reference table replication, cluster transparency
High Availability (HA)	K-Safety, network fault detection, live node rejoin, snapshots, command logging
Durability	Snapshots and command logging for database crash recovery
Database Replication	Cluster-wide replication for disaster recovery, hot stand-by and workload optimization
Realtime analytics	Materialized views and distributed read optimizations
Provisioning, management & monitoring	VoltDB Enterprise Manager and REST management API
VoltDB Studio for rapid application development	Companion tool for popular IDEs that supports rapid iteration of dev/test/tuning cycle
Data integration	Streaming export (including buffering and buffer overflow), and Hadoop integration
Developing applications with VoltDB	VoltDB- and community-created client libraries, JDBC and sample reference implementations

1. High-Throughput, Low-Latency SQL Database Operations

The following architectural innovations enable VoltDB to process millions of operations per second with very low latency:

In-memory Operation

Today's standard, off-the-shelf servers can be equipped with hundreds of gigabytes of main memory; there's no longer a need for DBMS performance to be limited by spinning disks. VoltDB stores data in memory and provides access to that data at memory speeds. In-memory processing eliminates disk waits from within VoltDB transactions, along with the need for associated latching and buffer management.

Stored Procedure Interface

Database access calls that travel the network between the application and the database can slow performance and increase latencies. VoltDB eliminates this overhead via a high performance stored procedure interface, so there is only ***one*** round trip between the client and server per transaction. In VoltDB, each stored procedure is defined as a transaction. The stored procedure (i.e., transaction) succeeds or rolls back as a whole, ensuring database consistency.

Traditional: `salary := get_salary(employee_id);`

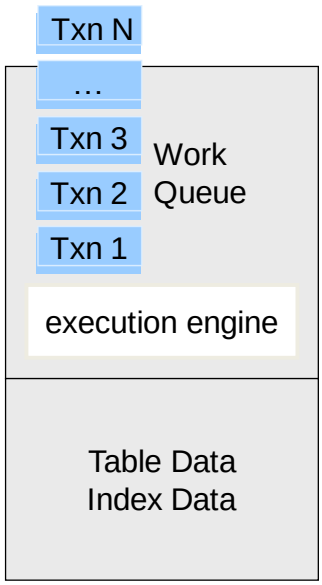
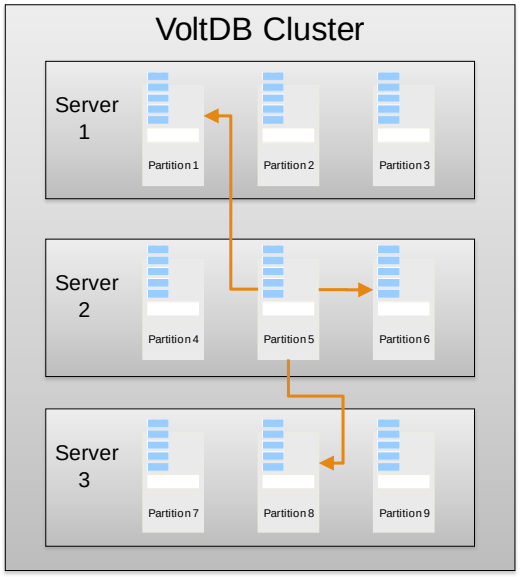
VoltDB: `callProcedure(asyncCallback, "get_salary", employee_id);`

Asynchronous/Synchronous Stored Procedure Execution

Conventional databases experience disk and user stalls within transactions. Rather than allow the CPU to be idle during the stalls, those DBMSs interleave SQL execution from multiple transactions during the waits so the CPU is always busy. This activity causes significant 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). VoltDB thereby eliminates the overhead associated with multi-threaded latching and locking.

A VoltDB database is composed of many in-memory execution engines called "partitions." A partition combines data and associated processing constructs. VoltDB automatically creates and distributes these to the CPU cores in the cluster. Each VoltDB "site" is single-threaded and contains a queue of transaction requests, which it executes sequentially — and exclusively — against its data.

Single Partition Transactions	Multiple Partition Transactions
	
Each VoltDB partition operates autonomously, freeing the rest of the cluster to handle other requests in parallel.	When a procedure requires data from multiple partitions, one node acts as a coordinator and distributes the necessary work to the other nodes, collects the results and completes the task. Transactional integrity is maintained and the architecture of multiple parallel partitions ensures throughput is kept at a maximum.

By using serialized processing, VoltDB ensures transactional consistency without the overhead of locking, latching, and transaction logs, while partitioning lets the database handle multiple requests in parallel. By eliminating stalls via in-memory operations and stored procedure calls, VoltDB is able to execute millions of SQL operations per second.

2. VoltDB Scaling Architecture

A VoltDB database can be scaled bi-directionally – by increasing the capacity of existing cluster nodes (scaling "up") and by increasing the number of nodes in a cluster (scaling "out"). Scaling up strategies leverage VoltDB's raw node speed properties and allows some multi-partition transactions to execute at minimal (bus) latencies. Scaling out strategies allow database throughput to be increased in linear fashion at the lowest possible costs.

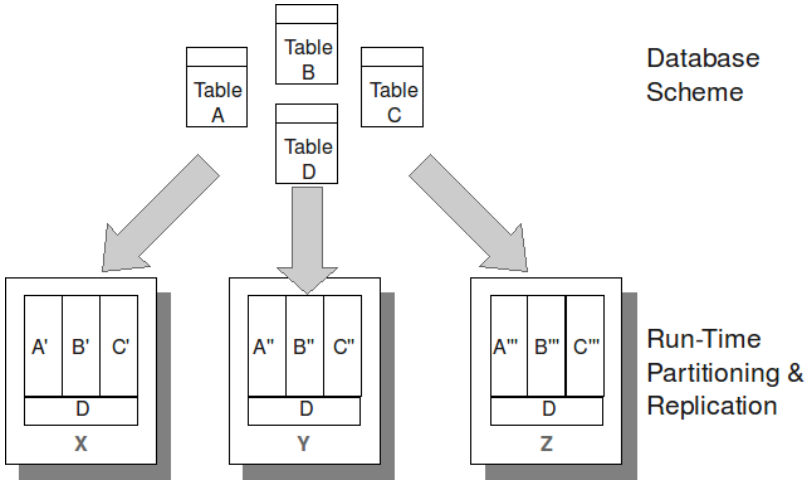
> *"VoltDB is very scalable; it should scale to 120 partitions, 39 servers, and 1.6 million complex transactions per second at over 300 CPU cores."*
>
> *Baron Schwartz*
> *Chief Performance Architect*
> *Percona*

Regardless of approach, scaling a VoltDB database increases the number of database partitions and execution queues. Scaling doesn't require any changes to the database schema or application code, nor does it require replacing existing hardware. As a general rule of thumb, the more processing cores (and therefore the more partitions) in the cluster, the more transactions VoltDB completes per second, providing an easy, linear path for handling a wide range of capacity and performance requirements.[1]

Partitioned versus Replicated Tables

Tables are partitioned in VoltDB by hashing primary key values. Performance is optimized by choosing partitioning keys that match the way the data is most commonly accessed.

To further optimize performance, VoltDB allows selected tables to be replicated on all partitions of the cluster. This strategy minimizes cross-partition join operations. For example, a retail merchandising database that uses product codes as the primary key may have one table that simply correlates the product code with the product's category and full name. Since this table is relatively small and does not change frequently (unlike inventory and orders) it can be replicated to all partitions. This way stored procedures can retrieve and return user-friendly product information when searching by product code without impacting the performance of order and inventory updates and searches.



Partitioning is appropriate for large tables (A, B, C) while replication of small, read-mostly tables (D) improves performance.

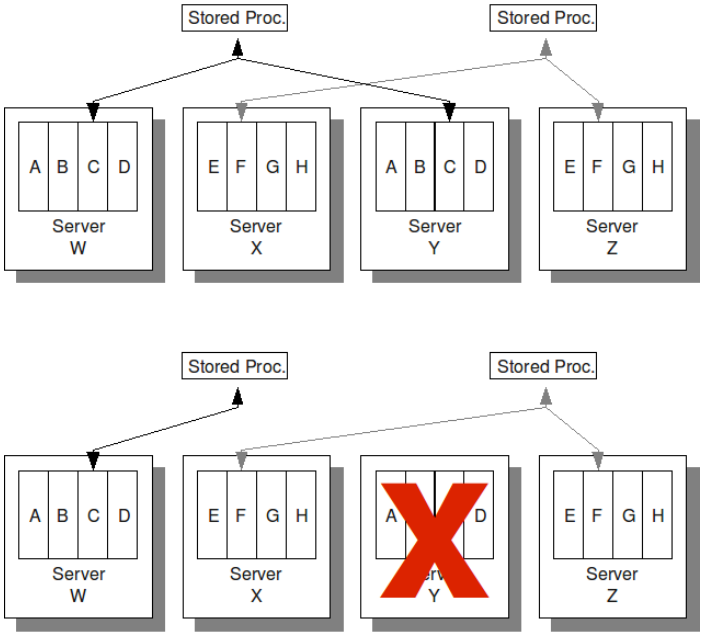
[1] Scalability benchmarks: http://www.mysqlperformanceblog.com/2011/02/28/is-voltdb-really-as-scalable-as-they-claim/

3. High Availability (HA)

Unlike most RDBMS products, which rely on third party solutions or expensive add-on features for HA, VoltDB was designed from the ground up to provide "Tandem-style" fault tolerance. VoltDB's HA solution consists of three related capabilities: K-safety, network fault detection and live node rejoin.

K-safety

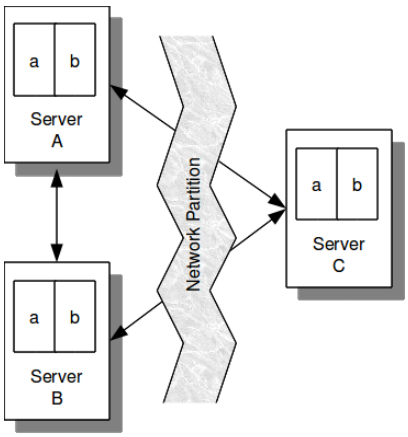
Technically, VoltDB delivers fault tolerance through an innovative synchronous multi-master replication strategy called "K-safety". When a database is configured for K-safety, VoltDB automatically (and transparently) replicates database partitions so that the database can withstand the loss of "K" nodes (due to hardware or software problems) without interrupting the database. For example, a K value of zero means that there is no replication; losing any server node will stop the database. If there are two copies of every partition (a K value of one), then the cluster can withstand the loss of at least one node (and possibly more) without interruption in service. The replicated partitions are fully functioning members of the cluster, including all read and write operations that apply to those partitions. In other words, the replicas function as peers rather than in a master-slave relationship.



During normal operations all work is executed synchronously on all replicas. If a node fails, the database continues to function by executing work on the remaining replicas.

Network Fault Detection

When a network partition occurs (e.g., a network switch fails), VoltDB nodes may become physically disconnected from each other. Each side of the partition "thinks" the nodes on the other side have gone down. With K-safety enabled, it's possible that both of the sub clusters have a complete copy of the database and are technically able to continue operations. This "split brain" scenario can cause significant data synchronization problems if it's not immediately detected.



VoltDB automatically detects network faults and takes appropriate remedial steps to eliminate split-brain operations. When VoltDB

detects a network fault condition, it immediately evaluates which side of the partition should continue operations and assigns all work to the "surviving" sub cluster. VoltDB also automatically snapshots the data in the "orphaned" sub cluster and performs an orderly shutdown of those nodes. Once the network partition has been repaired, the orphaned nodes can be reintroduced to the cluster, without the need for a service window, using the Live Node Rejoin feature described below.

Live Node Rejoin

VoltDB nodes that have been taken offline (due to system failures or scheduled maintenance) can be reintroduced to the running cluster via the "rejoin" operation. As a node first rejoins the cluster, it retrieves a copy of the data for its partitions from its sibling nodes. While the rejoined node is acquiring data and updates, its siblings continue to service database requests in a normal manner (although sometimes at slightly lower performance, since they're also doing work to bring the rejoining node up to date). Once the rejoined node catches up with its siblings, it returns to normal operation and begins accepting work, returning the cluster to its full K-safety and performance status.

4. Durability

While VoltDB's HA capabilities significantly reduce the probability of database downtime, severe hardware and software failures sometimes do occur. Recovery from systematic failures must happen quickly and accurately. Also, systems administrators sometimes need to perform orderly database shutdowns for scheduled maintenance. VoltDB provides high performance snapshotting and command logging to support a wide range of database durability requirements.

Snapshots

At defined intervals, VoltDB writes database snapshots to disk. Snapshot files can be used to restore the database to a previous, known state after a failure or scheduled service event. Snapshots are guaranteed to be transactionally consistent at the point in time at which the snapshot was completed. Importantly, snapshots pose minimal performance penalties on the database and can be configured to run at scheduled intervals, including "continuously".

Command Logging

Applications requiring 100% database recoverability can be easily configured to perform VoltDB command logging – persistent storage of every stored procedure invocation that occurs between database snapshots. In the event of a severe failure, the database can be recovered by restoring snapshots and replaying the command logs (which VoltDB does automatically when the database is restarted). VoltDB's command logging feature is highly configurable, allowing system administrators to choose the style of logging (synchronous or asynchronous), and the frequency with which command logs are flushed to disk (fsynch). Thus, VoltDB databases can be configured to deliver the optimal balance between throughput, latency and recoverability.

5. Database Replication

To complement HA and durability, VoltDB includes commercial-grade database replication capabilities that allow VoltDB databases to be automatically replicated within and across data centers. Available in the VoltDB Enterprise Edition, Database Replication ensures that every database transaction applied to a master VoltDB database is asynchronously applied to a defined replica database. Following a catastrophic crash, the database replica can be quickly and easily promoted to be the master, and all traffic can be redirected to that cluster. Once the original master has been recovered, the process can be easily reversed.

In addition to serving disaster recovery needs, Database Replication can also be used to maintain a hot standby database (i.e., to eliminate service windows while doing systems maintenance) and for workload optimization where, for example, write traffic is directed to the master VoltDB database, and read traffic is directed to the replica.

6. Realtime Analytics

VoltDB can be used to reveal insights, trends and anomalies as they occur in real time. Advanced VoltDB features like materialized views and optimized distributed reads allow data to be aggregated and enriched to serve a variety of analysis, reporting and alerting needs. VoltDB integrates easily with popular business intelligence tools to enable informative, realtime dashboards.

Distributed Read Operations

Some distributed datastores are inefficient at processing global read and summary operations. VoltDB applies MapReduce-style processing to many distributed query plans, executing most work at each partition. Thus, queries such as "`SELECT COUNT(*), SUM(Total) FROM OrderTable`" run quickly and efficiently in VoltDB, even though OrderTable may be extensively partitioned across the cluster.

Materialized Views

VoltDB's materialized view feature offers the ability to maintain realtime aggregates, counters and other interesting data derivations. Materialized views are useful for maintaining leader boards, performing realtime pattern analysis and detection, and driving realtime monitoring events and alerts. The CREATE VIEW statement is used to define a materialized view of a table with selected columns and aggregates. The view is stored as a special table in the database and is maintained – *without significant performance costs* – each time the underlying database table is updated. Because the view is materialized, SELECTs on it will execute extremely efficiently.

7. Provisioning, Management and Monitoring

VoltDB provides enterprise-class tools and APIs to facilitate the provisioning, monitoring and management of VoltDB database clusters.

VoltDB Enterprise Manager (VEM)

The VEM is a browser based console that allows people to manage VoltDB clusters from anywhere. Using the VEM, there is no need to pre-install VoltDB on each node. The management console handles distributing both the VoltDB software and the database catalogs to the cluster nodes. The console also helps by providing a common interface for:

- Initiating and collecting snapshots
- Providing live statistics on database volume, latency, memory utilization and performance
- Comparing and updating the database catalog and schema
- Administering (dropping and rejoining) nodes in a highly available cluster

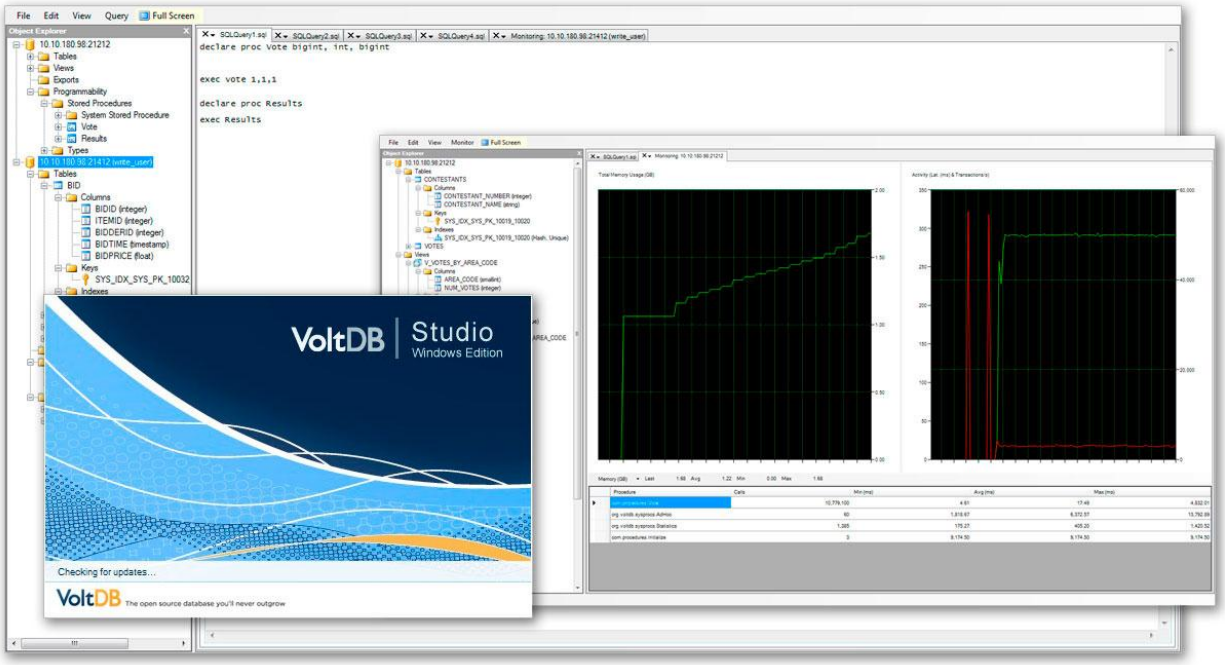
REST Management API

In addition to the VEM's graphical interface, VoltDB exposes a REST ("Representational State Transfer") programming interface, allowing database operations personnel to script any of the VEM functions for further customization and simplification of the common database administration tasks.

8. VoltDB Studio for Rapid Application Development

VoltDB Studio is a browser-based environment that supports rapid development, testing and tuning of VoltDB applications. Design to be used as a companion to popular IDEs, VoltDB Studio provides visual tools for assisting most common development and test activities, including:

- Connecting to VoltDB databases
- Browsing schema and system objects
- Running ad-hoc queries and testing stored procedures
- Performing pre-deployment database performance analysis and tuning
- Browsing useful code snippets



VoltDB Studio is included in all VoltDB database distributions, and it's accessible directly from within the VoltDB Demo Dashboard application (discussed in the Getting Started section below).

9. Data Integration

In addition to supporting realtime analytics (see Section 5 above), VoltDB seamlessly integrates with OLAP and Hadoop infrastructures, feeding data to those environments for deep analytical exploration. This section discusses some of the VoltDB features that make integration fast and easy.

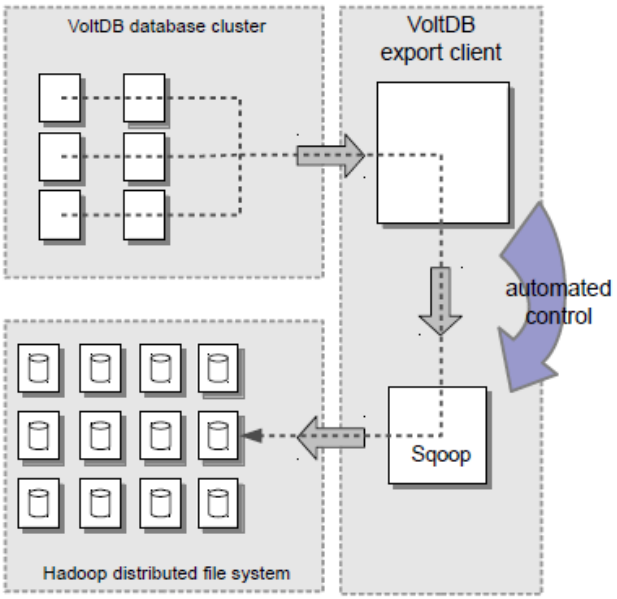
Streaming Exports

VoltDB is able to selectively export data as it is committed to the database. The target for exporting data from VoltDB may be another database, a file (such as a sequential log), or a process (such as a systems monitor). VoltDB enables streaming exports via "Export Tables," special tables containing data from columns from one or more physical tables in the database. As soon as data is inserted into the Export Table, the data is sent through the connector to a receiving application.

VoltDB holds the Export Table data in a buffer until the export client fetches it. If the export client does not keep up with the connector and the data queue fills up, VoltDB writes overflow data to disk to ensure that all exported data is available to the client when it eventually resumes fetching. Export overflow eliminates "impedance mismatch" situations that could occur between VoltDB and downstream systems, where VoltDB is streaming data faster than the companion database can ingest.

Hadoop Integration

VoltDB is an excellent solution for handling data in high velocity state; Hadoop is an excellent solution for analyzing massive volumes of historical data. Thus, the combination of VoltDB and Hadoop offer the flexibility to handle a continuum of "fast" and "deep" data applications.



VoltDB comes with an export client that automates the process of exporting data from VoltDB to Hadoop. VoltDB can integrate directly with HDFS or through Hadoop's Sqoop import technology. VoltDB's standard export client collects data from the VoltDB cluster, de-serializes and buffers that data for downstream delivery. The VoltDB export client initiates downstream load processes and delivers data to them, after which data is loaded into the target Hadoop datastore.

10. Developing Applications with VoltDB

VoltDB includes client libraries that support native database access from a number of popular programming languages. Many of the VoltDB client libraries are developed and maintained by the VoltDB engineering team, but some are also contributed by VoltDB open source community members.

VoltDB-Provided Libraries		Community-Developed Libraries
Java C++ .NET (C#)	Python PHP HTTP/JSON	Erlang Ruby Node.js

JDBC Interface

In addition to the client libraries discussed above, VoltDB 2.0 (and higher) supports the JDBC API. VoltDB's JDBC interface allows users to process ad hoc SQL; it supports execution of prepared plans; and it provides an alternative interface for invoking VoltDB stored procedures. The JDBC interface provides standard return codes and also implements JDBC metadata classes for retrieval of database catalog metadata.

Sample Apps and Reference Implementations

VoltDB also includes a growing collection of sample applications and reference implementations. These resources are aimed at helping developers get up to speed and begin building VoltDB applications quickly and, therefore, are provided with full source code.

- **Voter** – a sample application patterned after a well-known TV talent show contest, Voter illustrates the basics of database partitioning, high volume write operations and realtime analytics.

- **VoltKV** – a simple Key/Value interface for VoltDB, VoltKV illustrates how VoltDB can be used to support applications that need the combined benefits of schema flexibility, high write throughput and ACID transactions.

- **VoltCache** – a reference implementation that expands on the Key/Value store concept, providing a memcached-like interface with support for data expiration, counters and advanced operations. Because it leverages VoltDB's architecture, VoltCache is a no-hassle scale-out caching model that eliminates node synchronization requirements and yields consistent answers, throughput and latency to all client applications.

11. Getting Started

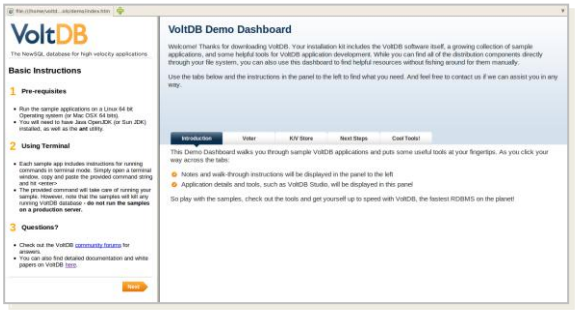
VoltDB is available in two distributions – Community Edition and Enterprise Edition. Community Edition is open source (GPL3) and geared toward developers who are building, testing and tuning VoltDB applications. Enterprise Edition is provided under VoltDB's commercial license and is intended for organizations that are deploying VoltDB applications into production. VoltDB runs natively on supported 64-bit Linux systems. Developers can also build VoltDB applications on 64-bit Mac OSX. VoltDB's main software downloads page, including client libraries and documentation, can be found here. Pre-packaged evaluation versions of VoltDB are available for Amazon EC2 and VMWare:

VoltDB can be setup to run on a cluster of choice on Amazon EC2 within just a few minutes. VoltDB for Amazon EC2 includes the VoltDB Demo Dashboard, VoltDB Studio, sample applications and other developer resources. Get it here.

The VoltDB VMware image can be executed using a variety of VMware players. Although VoltDB is a 64-bit Linux application, VMware allows it to execute as a virtualized image on other platforms such as Windows. Get it here.

VoltDB Demo Dashboard

The VoltDB Demo Dashboard is bundled with all VoltDB distributions. It's a useful tool for product evaluators and developers who are new to VoltDB, providing convenient access to a variety of introductory resources, including sample apps, reference implementations and VoltDB Studio from a simple browser based interface. Source code for most resources in the Demo Dashboard is readily accessible in the VoltDB installation directory.



To learn more about VoltDB, visit www.voltdb.com. The links below also provide quick access to the most popular technical content on our website. To discuss your high performance database needs with a VoltDB engineer, please call Sales at +1.978.528.4660 or send us an email.