VoltDB: an SQL Developer's Perspective

Tim Callaghan, VoltDB Field Engineer
tcallaghan@voltdb.com

Agenda

- VoltDB Technical Overview
- Comparing VoltDB to Traditional OLTP
- Migration and Development
- Q+A

About Me

- VoltDB Field Engineer/Community Advocate
  - Joined VoltDB in September, 2009
  - Support commercial and community customers
  - Built many examples and POCs
- Technical background
  - 18 years of Oracle design/development/administration
  - User of many programming languages – database and traditional
- See last slide for full contact information

VoltDB Technical Overview

Before I Begin…

VoltDB is available in community and commercial editions

Runs on Linux + Mac*

Scaling Traditional OLTP Databases

- Sharding improves performance but introduces…
  - Management complexity
    - + disjointed backup/recovery and replication
    - + manual effort to re-partition data
  - Application complexity
    - + shard awareness
    - + cross partition joins
    - + cross partition transactions
  - And, each shard still suffers from traditional OLTP performance limitations
- If you can shard, your application is probably great in VoltDB.

Technical Overview

- "OLTP Through the Looking Glass"

  http://cs-www.cs.yale.edu/homes/dna/papers/oltpperf-sigmod08.pdf

- VoltDB avoids the overhead of traditional databases
  - K-safety for fault tolerance

  - no logging

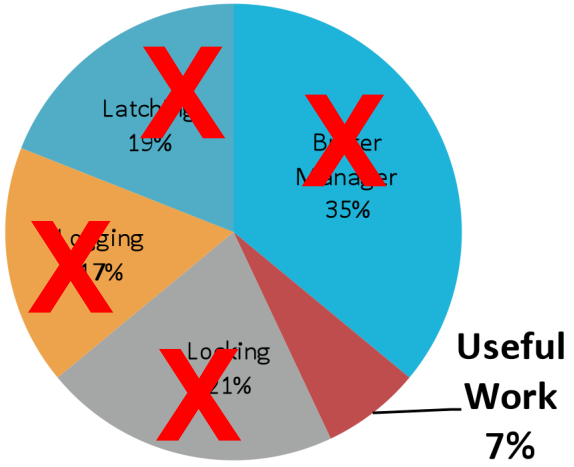
  - In memory operation for maximum throughput

  - no buffer management

  - Partitions operate autonomously and single-threaded

  - no latching or locking

- Built to horizontally scale

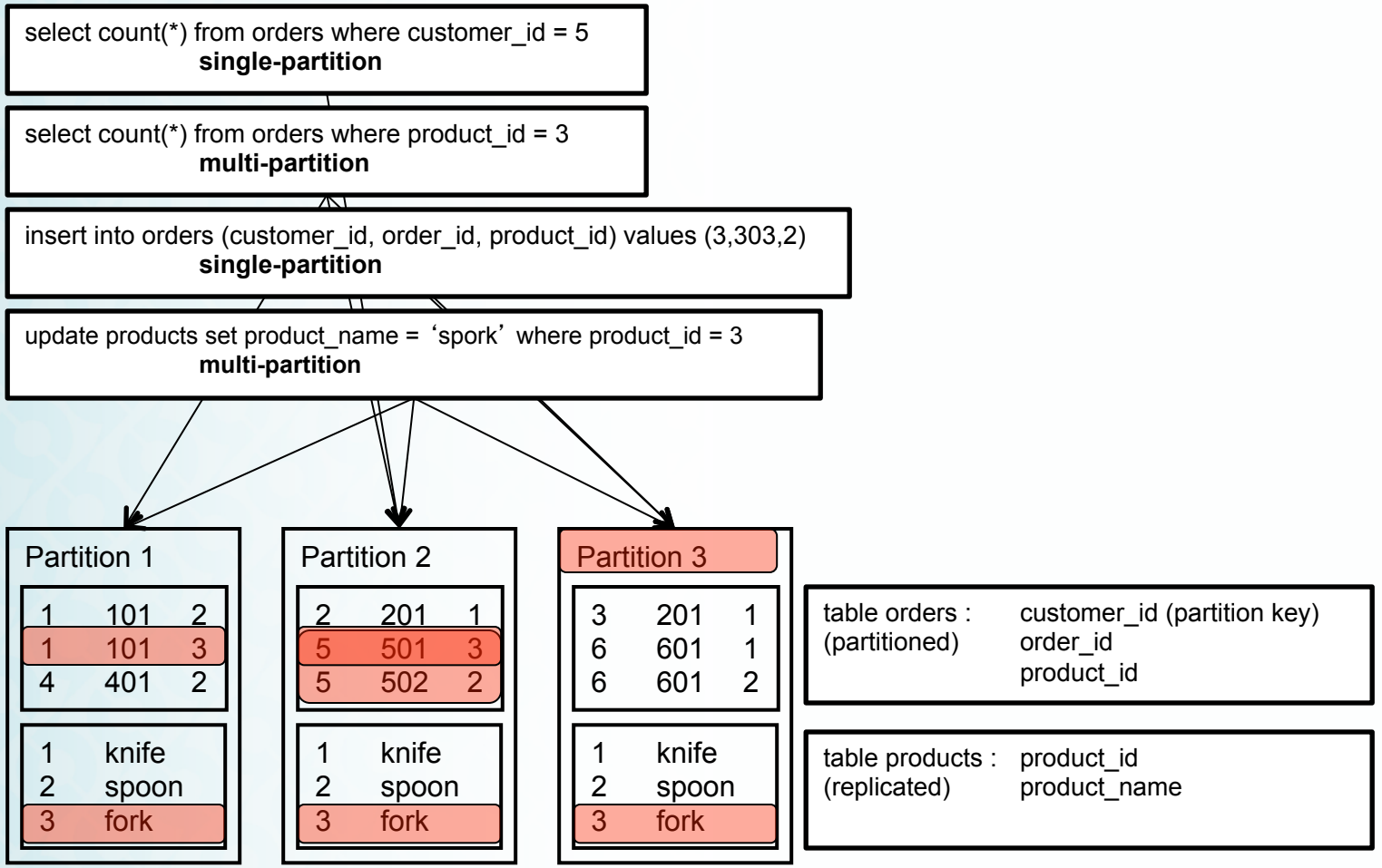


Technical Overview – Partitions (1/3)

- 1 partition per physical CPU core
  - Each physical server has multiple VoltDB partitions
- Data - Two types of tables
  - Partitioned
    - + Single column serves as partitioning key
    - + Rows are spread across all VoltDB partitions by partition column
    - + Transactional data (high frequency of modification)
  - Replicated
    - + All rows exist within all VoltDB partitions
    - + Relatively static data (low frequency of modification)
- Code - Two types of work – both ACID
  - Single-Partition
    - + All insert/update/delete operations within single partition
    - + Majority of transactional workload
  - Multi-Partition
    - + CRUD against partitioned tables across multiple partitions
    - + Insert/update/delete on replicated tables

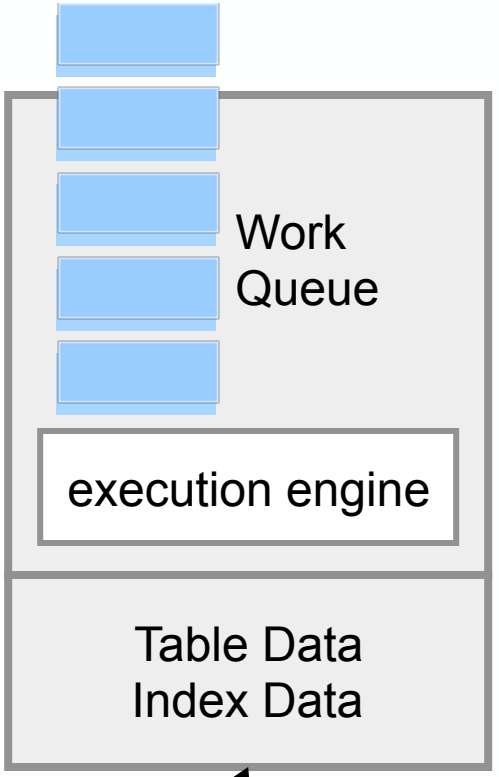
Technical Overview – Partitions (2/3)

- **Single-partition vs. Multi-partition**



Technical Overview – Partitions (3/3)

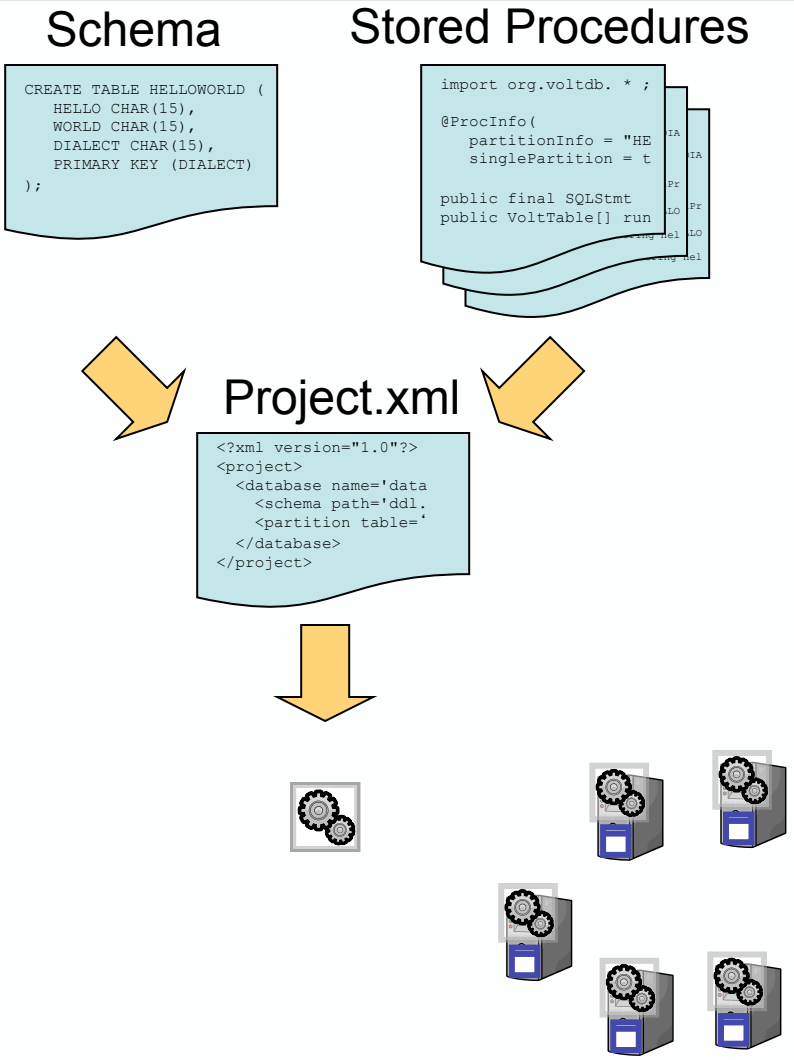
- Looking inside a VoltDB partition…
  - Each partition contains data and an execution engine.
  - The execution engine contains a queue for transaction requests.
  - Requests are executed sequentially (single threaded).



- Complete copy of all replicated tables
- Portion of rows (about 1/partitions) of all partitioned tables

Technical Overview – Compiling

- The database is constructed from
  - The schema (DDL)
  - The work load (Java stored procedures)
  - The Project (users, groups, partitioning)
- VoltCompiler creates application catalog
  - Copy to servers along with 1 .jar and 1 .so
  - Start servers



Technical Overview - Transactions

- All access to VoltDB is via Java stored procedures (Java + SQL)
- A single invocation of a stored procedure is a transaction (committed on success)
- Limits round trips between DBMS and application
- High performance client applications communicate asynchronously with VoltDB

Technical Overview – Clusters/Durability

- Scalability
  - Increase RAM in servers to add capacity
  - Add servers to increase performance / capacity
  - Consistently measuring 90% of single-node performance increase per additional node
- High availability
  - K-safety for redundancy
- Snapshots
  - Scheduled, continuous, on demand
- Spooling to data warehouse
- Disaster Recovery/WAN replication (Future)
  - Asynchronous replication

Comparing VoltDB to Traditional OLTP

(in no particular order)

Asynchronous Communications

- Client applications communicate asynchronously with VoltDB
  - Stored procedure invocations are placed “on the wire”
  - Responses are pulled from the server
  - Allows a single client application to generate > 100K TPS
  - Our client library will simulate synchronous if needed

Traditional

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

VoltDB

```
callProcedure(asyncCallback, “get_salary”, employee_id);
```

Transaction Control

- VoltDB does not support client-side transaction control
  - Client applications cannot:
    - insert into t_colors (color_name) values (‘purple’);
    - rollback;
  - Stored procedures commit if successful, rollback if failed
  - Client code in stored procedure can call for rollback

Interfacing with VoltDB

- Client applications interface with VoltDB via stored procedures
  - Java stored procedures – Java and SQL
  - No ODBC/JDBC

Lack of concurrency

- Single-threaded execution within partitions (single-partition) or across partitions (multi-partition)
- No need to worry about locking/dead-locks
  - great for “inventory” type applications
    + checking inventory levels
    + creating line items for customers
- Because of this, transactions execute in microseconds.
- However, single-threaded comes at a price
  - Other transactions wait for running transaction to complete
  - Don’t do anything crazy in a SP (request web page, send email)
  - Useful for OLTP, not OLAP

Throughput vs. Latency

- VoltDB is built for throughput over latency
- Latency measured in mid single-digits in a properly sized cluster
- Do not estimate latency as (1 / TPS)

SQL Support

- SELECT, INSERT (using values), UPDATE, and DELETE
- Aggregate SQL supports AVG, COUNT, MAX, MIN, SUM
- Materialized views using COUNT and SUM
- Hash and Tree Indexes
- SQL functions and functionality will be added over time, for now I do it in Java
- Execution plan for all SQL is created at compile time and available for analysis

SQL in Stored Procedures

- SQL can be parameterized, but not dynamic

"select * from foo where bar = ?;" (YES)

"select * from **?** where bar = ?;" (NO)

Connecting to the Cluster

- Clients connect to one or more nodes in the VoltDB cluster, transactions are forwarded to the correct node
  - Clients are not aware of partitioning strategy
  - In the future we may send back data in the response indicating if the transaction was sent to the correct node.

Schema Changes

- Traditional OLTP
  - add table…
  - alter table…
- VoltDB
  - modify schema and stored procedures
  - build catalog
  - deploy catalog
- V1.0: Add/drop users, stored procedures
- V1.1: Add/drop tables
- Future: Add/drop column, …

Table/Index Storage

- VoltDB is entirely in-memory
- Cluster must collectively have enough RAM to hold all tables/indexes (k + 1 copies)
- Even data distribution is important

Migration and Development

Migrating to VoltDB

1. Grab your existing DDL, SQL, and stored procedures.
2. Compile your VoltDB Application.
3. Sorry…

- Review your application as a whole
- Partitioning is HUGE
- Everything inside stored procedures
- SQL requirements

Client Libraries

- We support Java and C++ natively
- PHP is via C++/SWIG
- SWIG can produce many others
- Long-term roadmap is native PHP, Python, C#, and others.
- Community has developed Erlang (wire protocol) and Ruby (HTTP/JSON)
- HTTP/JSON interface also available
  - Easily used by most languages
  - Server can handle ~1,000 requests per second

Getting Data In and Out

- In: Create simple client application to read flat file and load into table(s).
  - 1 SQL stored procedures can be defined in XML
  - Working on a generic utility for loading
- Out: Snapshot to flat file
  - Snapshots can be converted to CSV/TSV data
- Out: EL
  - Special type of tables in VoltDB
    - + export (insert/update/delete) or export only (insert only)
    - + client application reads from buffers and acks when done
      - currently targeting file-system, JDBC coming

Q & A

- Visit http://voltdb.com to…
  - Download VoltDB
  - Get sample app code
- Join the VoltDB community
  - VoltDB user groups: www.meetup.com/voltdb
  - Follow VoltDB on Twitter @voltdb
- Contact me
  - tcallaghan@voltdb.com (email)
  - @tmcallaghan (Twitter)