Erlvolt: Scaling ACID

Erlang Factory San Francisco 2014

H. Diedrich

http://www.eonblast.com – twitter @hdiedrich to Gustav

https://github.com/Eonblast/Erlvolt

The Quest for Perfection

ME TO THE 30TH CENTURY

studio that **pushes the**
what can be achieved in games.

level of browser based Massively
Games (MMOG), both **visually**,
technologically. Then put
P) back into the game and you're

wer of the **Cloud** crossed with emerging
n of **Alternate Reality** games: players
o actually* **change their world**.

bining the crafts of **games** and **film** to
a new, **more immersive experience**.

pposed to make believe.

RK OF EONBLAST CORPORATION.

in Game Servers.

Your Host

Henning Diedrich

- Founder, CEO Eonblast
- Lead Software Engineer SumUp
- Maker of Erlvolt
- Former maintainer of Emysql

What is Erlvolt?

native Erlang VoltDB driver

Open Source

part of official distribution

used in production

https://github.com/Eonblast/Erlvolt

VoltDB and Erlang

Open Source Scale Dream Team

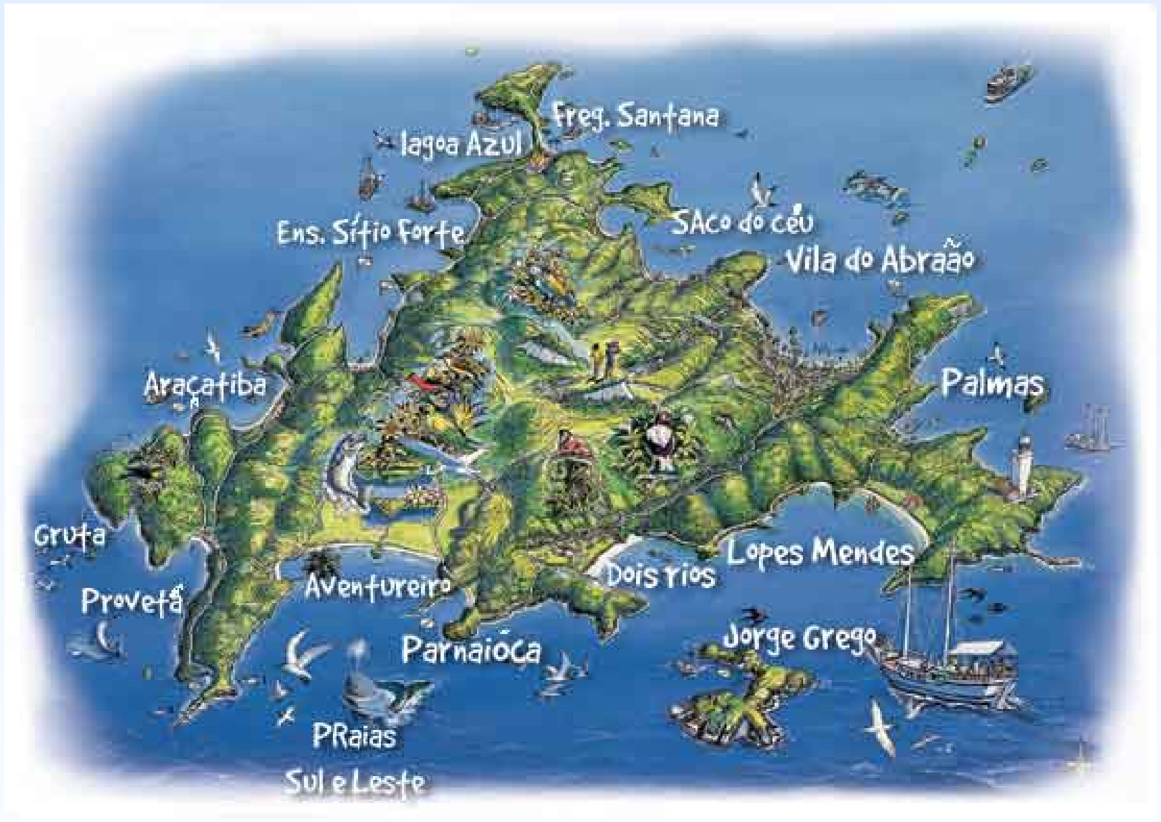
- How is it special?
- How does it look?
- Is it for me?

The Great Idea

Ilha Grande

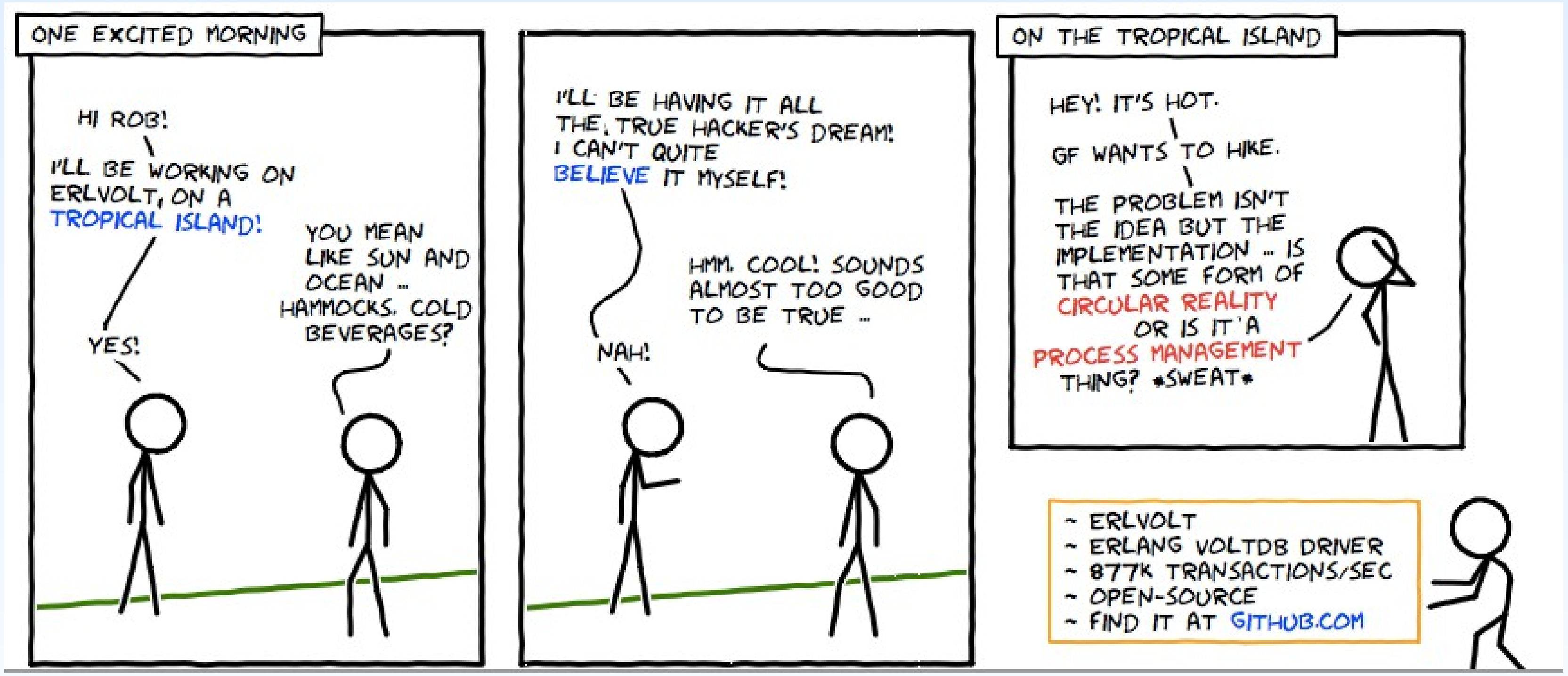
Erlvolt was in part programmed on a tropical island. Trying it takes only minutes: https://gist.github.com/hdiedrich/5415065

Pirate Island



The island looks like a pirate treasure island. In fact, it was. It is also quite humid and warm.

The Great Idea


ONE EXCITED MORNING
HI ROB!
I'LL BE WORKING ON ERLVOLT, ON A TROPICAL ISLAND!
YES!
YOU MEAN LIKE SUN AND OCEAN ... HAMMOCKS, COLD BEVERAGES?
I'LL BE HAVING IT ALL THE TRUE HACKER'S DREAM! I CAN'T QUITE BELIEVE IT MYSELF!
NAH!
HMM. COOL! SOUNDS ALMOST TOO GOOD TO BE TRUE ...
ON THE TROPICAL ISLAND
HEY! IT'S HOT.
GF WANTS TO HIKE.
THE PROBLEM ISN'T THE IDEA BUT THE IMPLEMENTATION ... IS THAT SOME FORM OF CIRCULAR REALITY OR IS IT A PROCESS MANAGEMENT THING? *SWEAT*
- ERLVOLT
- ERLANG VOLTDB DRIVER
- 877K TRANSACTIONS/SEC
- OPEN-SOURCE
- FIND IT AT GITHUB.COM

Xkcd-style courtesy http://cmx.io. Trying Erlvolt yourself takes only minutes: https://gist.github.com/hdiedrich/5415065

The Great Idea

- Horizontal Partitioning
- Single-Thread Execution
- Rhythmic Distribution

The Joe Armstrong of VoltDB

Mike Stonebraker

Ingres • PostgreSQL

"Time for a Complete Rewrite!"

Vertica • VoltDB

Paper: The End of an Architectural Era (It's Time for a Complete Rewrite) http://nms.csail.mit.edu/~stavros/pubs/hstore.pdf

What is VoltDB?

≈ Redis + SQL + Scale

In a nutshell, VoltDB could be described as a scalable Redis with SQL.

The Beauty

- In-Memory
- SQL
- ACID
- Replication
- Elasticity

Try Erlvolt

Install & run a local benchmark:

```
git clone -b voltdb-3.7.0.5 git://github.com/VoltDB/voltdb.git
git clone git://github.com/Eonblast/Erlvolt.git erlvolt
cd voltdb && ant && cd examples/voter && ./run.sh &
cd erlvolt && make profile bench
```

Trying it yourself takes only minutes including the full build. More details here: https://gist.github.com/hdiedrich/5415065

```
[...]

Initializing VoltDB...

 _    __      ____  ____  ____
| |  / /___  / / /_/ __ \/ __ )
| | / / __ \/ / __/ / / / __  |
| |/ / /_/ / / /_/ /_/ / /_/ /
|___/\____/_/\__/_____/_____/

--------------------------------

Build: 3.7 voltdb-3.7.0.5-0-g105a023-local Community Edition
Connecting to VoltDB cluster as the leader...
Host id of this node is: 0
WARN: Running without redundancy (k=0) is not recommended for production use.
Server completed initialization.
```

```
Erlvolt Bench 0.9 (client 'VSD')
---------------------------------------------------------------------------------------------
Client 'VSD', voter, 100,000 calls, steady, 200 workers,
delay n/a, direct, queue n/a, slots n/a, limit n/a, verbose, 5.0 stats/sec
Hosts: localhost:21212
connect ...
preparation ...
Start at: 2014-03-02 15:05:00 ................
Starting: 2014-03-02 15:05:00
calls ...

Client   VSD: at  0.203sec: lap    4,743 T/sec,  total    4,763 T/sec,
success:     967, fails: 0, pending:     200, avglat: 14.787ms, maxlat:   29ms

Client   VSD: at  0.400sec: lap    5,465 T/sec,  total    5,110 T/sec,
success:   2,044, fails: 0, pending:     200, avglat: 17.329ms, maxlat:   44ms

Client   VSD: at  0.604sec: lap    6,668 T/sec,  total    5,632 T/sec,
success:   3,402, fails: 0, pending:     103, avglat: 15.102ms, maxlat:   27ms
```

```
cool down ...
check writes ... ok
results ...  votes:     100,000 (6 contestants)
....Jessie Eichman:      16,812
......Kelly Clauss:      16,805
....Jessie Alloway:      16,717
...Tabatha Gehling:      16,598
.....Edwina Burnam:      16,567
.....Alana Bregman:      16,501
close pool ...
Client 'VSD', voter, 100,000 calls, steady, 200 workers,
delay n/a, direct, queue n/a, slots n/a, limit n/a, verbose, 5.0 stats/sec
---------------------------------------------------------------------------------------------
Client 'VSD' overall: 15,203 T/sec throughput, 0.00% fails,
total transactions: 100,000, fails: 0, total time: 6.577sec
Erlvolt 0.3.3, bench started 2014-03-02 15:05:00, ended 2014-03-02 15:05:06,
database: +100,000 new votes
[++++++++++++++++]
# 'make clean fast bench' for faster, HiPE-compiled beams.
# 'make clean profile bench' for rolling stats during bench.
```

Use Case?

Why VoltDB?

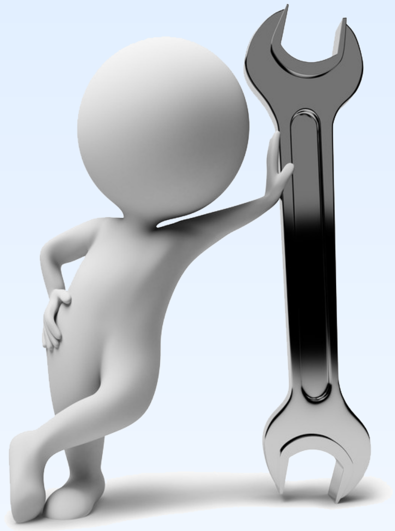
Business Perspective

- Big Data +
  real-time answers
- Reduced cost
- Strategic flexibility

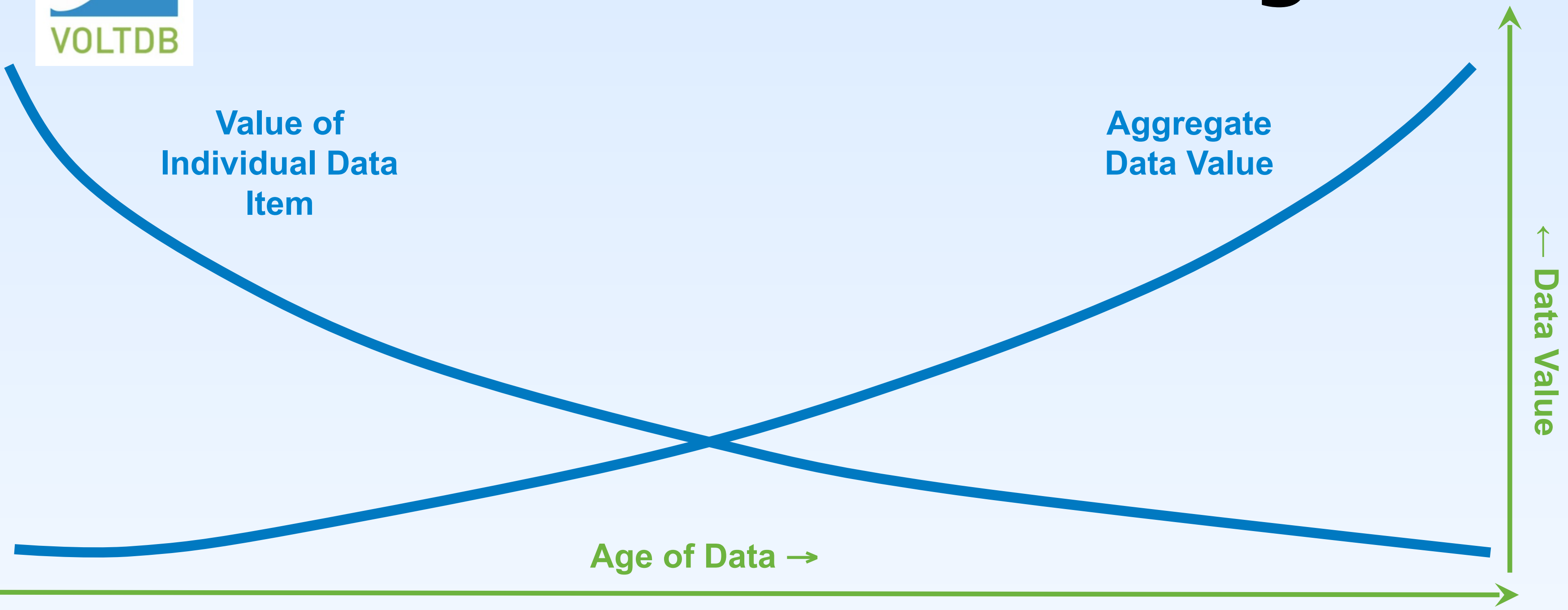
Why VoltDB?

Production Perspective

- The good parts of SQL
- Speed of in-memory
- Take pain out of scaling

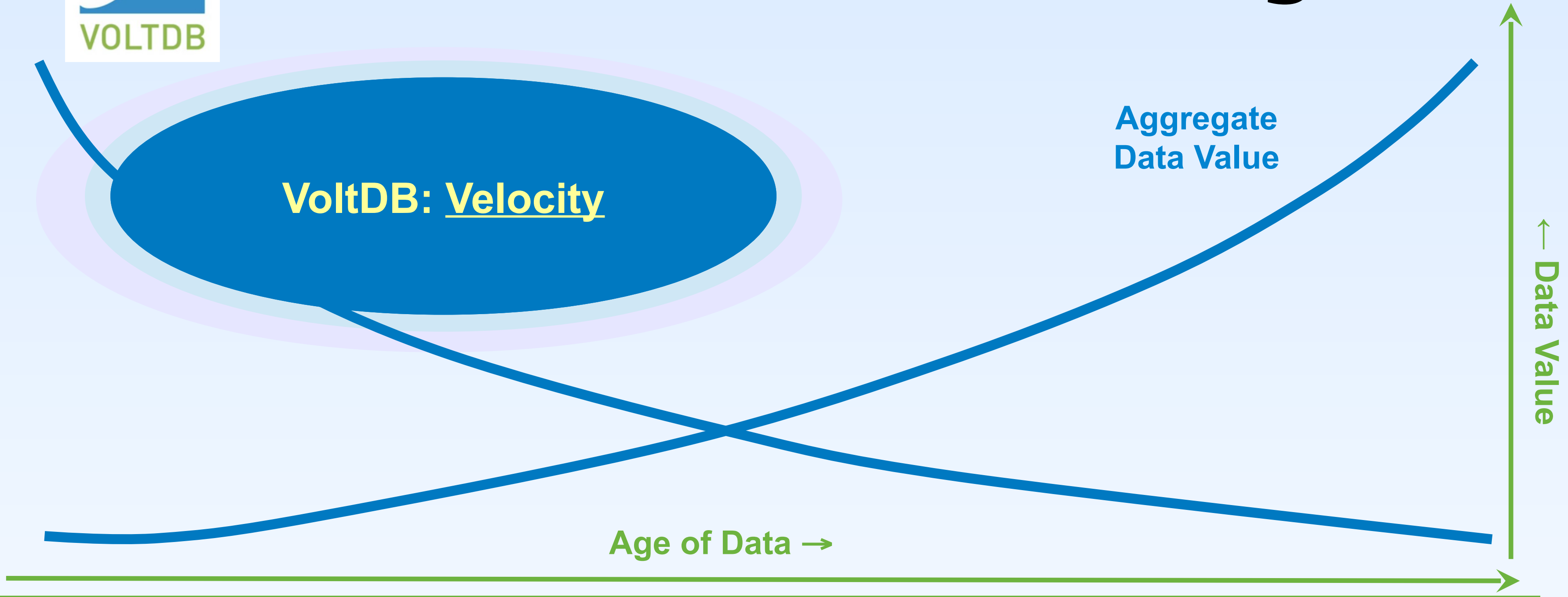


VoltDB Positioning



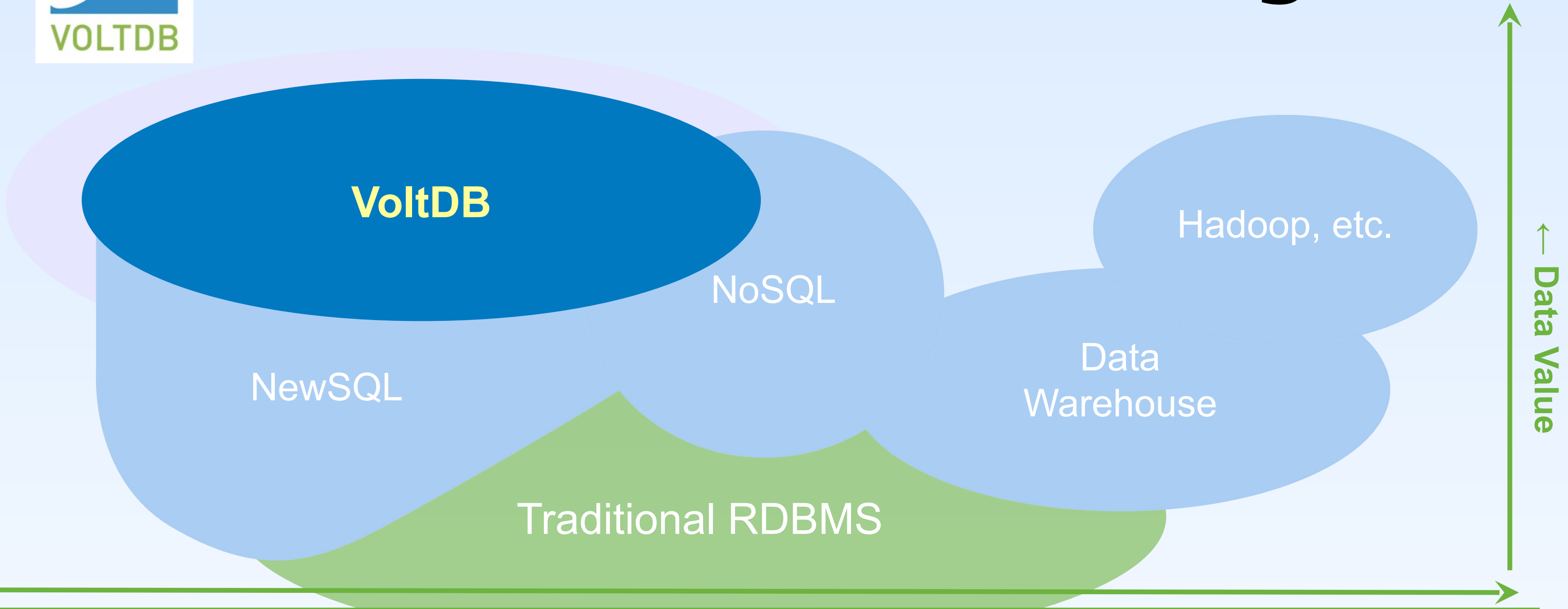
Interactive	Real-time Analytics	Record Lookup	Historical Analytics	Exploratory Analytics
Milliseconds	Hundredths of seconds	Second(s)	Minutes	Hours
· Place trade · Serve ad · Enrich stream · Examine packet · Approve trans.	· Calculate risk · Leaderboard · Aggregate · Count	· Retrieve click stream · Show orders	· Backtest algo · BI · Daily reports	· Algo discovery · Log analysis · Fraud pattern match

VoltDB Positioning



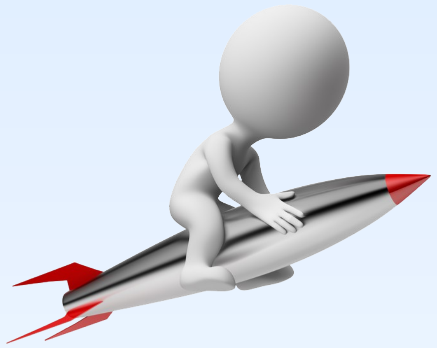
Interactive	Real-time Analytics	Record Lookup	Historical Analytics	Exploratory Analytics
Milliseconds	Hundredths of seconds	Second(s)	Minutes	Hours
· Place trade · Serve ad · Enrich stream · Examine packet · Approve trans.	· Calculate risk · Leaderboard · Aggregate · Count	· Retrieve click stream · Show orders	· Backtest algo · BI · Daily reports	· Algo discovery · Log analysis · Fraud pattern match

VoltDB Positioning



Interactive	Real-time Analytics	Record Lookup	Historical Analytics	Exploratory Analytics
Milliseconds	Hundredths of seconds	Second(s)	Minutes	Hours
· Place trade · Serve ad · Enrich stream · Examine packet · Approve trans.	· Calculate risk · Leaderboard · Aggregate · Count	· Retrieve click stream · Show orders	· Backtest algo · BI · Daily reports	· Algo discovery · Log analysis · Fraud pattern match

VoltDB Use Cases



- High throughput from relentless data feeds
- Fast operations on high value data
- Real-time analytics with immediate visibility
- Resilience to failure on commodity hardware

VoltDB Features

- ACID

  Full serializable isolation with strict atomicity and durability.
- SQL

  DML and DQL is SQL, embedded in Java Stored Procedures.
- HA

  Automatic synchronous intra-cluster and inter-datacenter replication.
- Scalable

  Horizontal shared-nothing clustering, 100,000+ TPS per node.

VoltDB Facts

- VoltDB, Inc. 2009 – commercial developer, support
- Open Source – 100% dictatorial by VoltDB, Inc
- Made for OLTP – fast cheap writes, high throughput
- CA of CAP – 100% consistent & highly available
- Simple SQL – real queries, indices, materialized views
- In-memory – 100x faster than MySQL
- ACID transactions – double bookkeeping
- Distributed – for painless growth
- Linear scale – predictable, low cost
- Elasticity – scale and repair on the fly
- Replication, Snapshots – disk persistence, hot backup
- More SQL than SQL – clean separation of data

- Made for a concrete need
- Made for distribution
- Made for multi-core
- Truly different approaches
- *Based* on hardware parallelism
- Improving on previous solutions
- Corporate-created
- Open Source
- Professional support
- Known by Those in the Know

175+ Customers

- **AOL's Games.com** using VoltDB as front end to its big data operation. Replaced NoSQL datastore that couldn't handle data velocity
- **Yahoo!** the home of Hadoop using VoltDB for high velocity data ingest and relational reporting
- **YellowHammer Media Group** pairing VoltDB with deep analytical database to create closed-loop systems where active user behaviors and historical data feed off each other to inform real-time decisions
- **Bursa Malaysia** deployed VoltDB to create previously impossible trading app that catches transaction errors in real time
- **Shopzilla** selected VoltDB over NoSQL and shared MySQL databases to connect shoppers in U.S. and Europe with over 100 million products from tens of thousands of retailers

The Beauty

In-Memory

"Redis of Clusters"

- In-Memory Speed
- Fully Distributed
- Fully Replicated
- Fully Disk persistent
- Good for 100s of GB of data

SQL

"The MySQL of NoSQLs"

- SQL has flaws, but it is:
- Essentially math, not syntax
- You could be missing queries
- VoltDB is 'more SQL than SQL'

CAP

- Distributed
- Consistent
- Highly-Available
- Partition-Tolerant

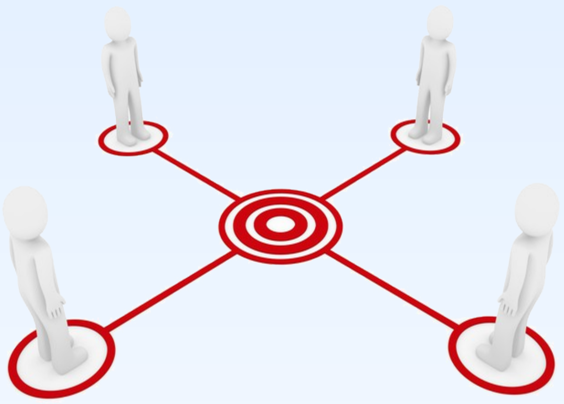
… have it all.

Brewer comes back to CAP in 2012: http://www.infoq.com/articles/cap-twelve-years-later-how-the-rules-have-changed

ACID

- Atomicity
- Consistency
- Isolation
- Durability

… a rare card.



Double Bookkeeping

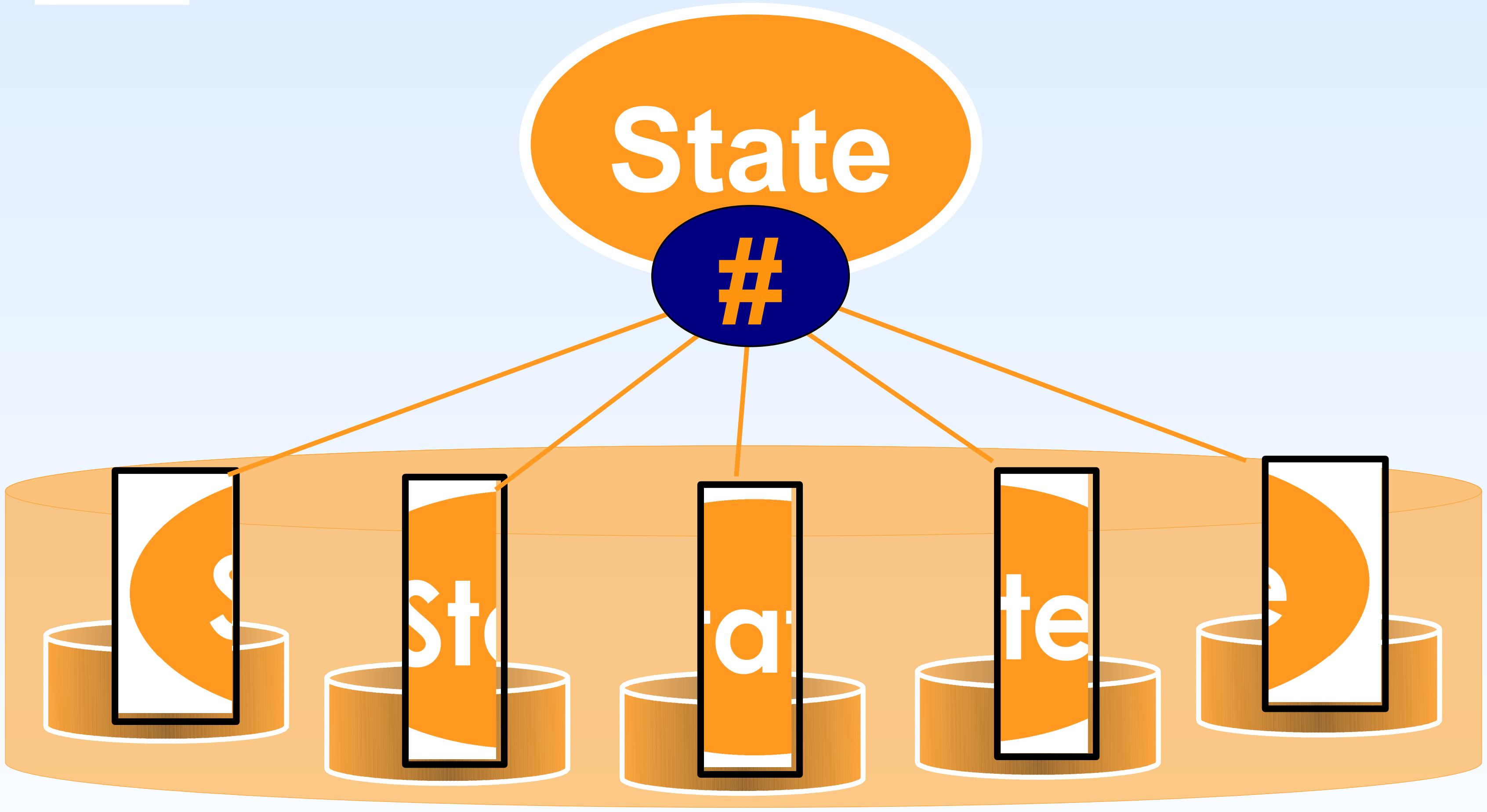
- Not Every Use Case needs It
- Requires ACID Transactions
- Neigh Impossible to emulate
- Impossible With BASE (Eventual Consistency)

The Magic

The Magic

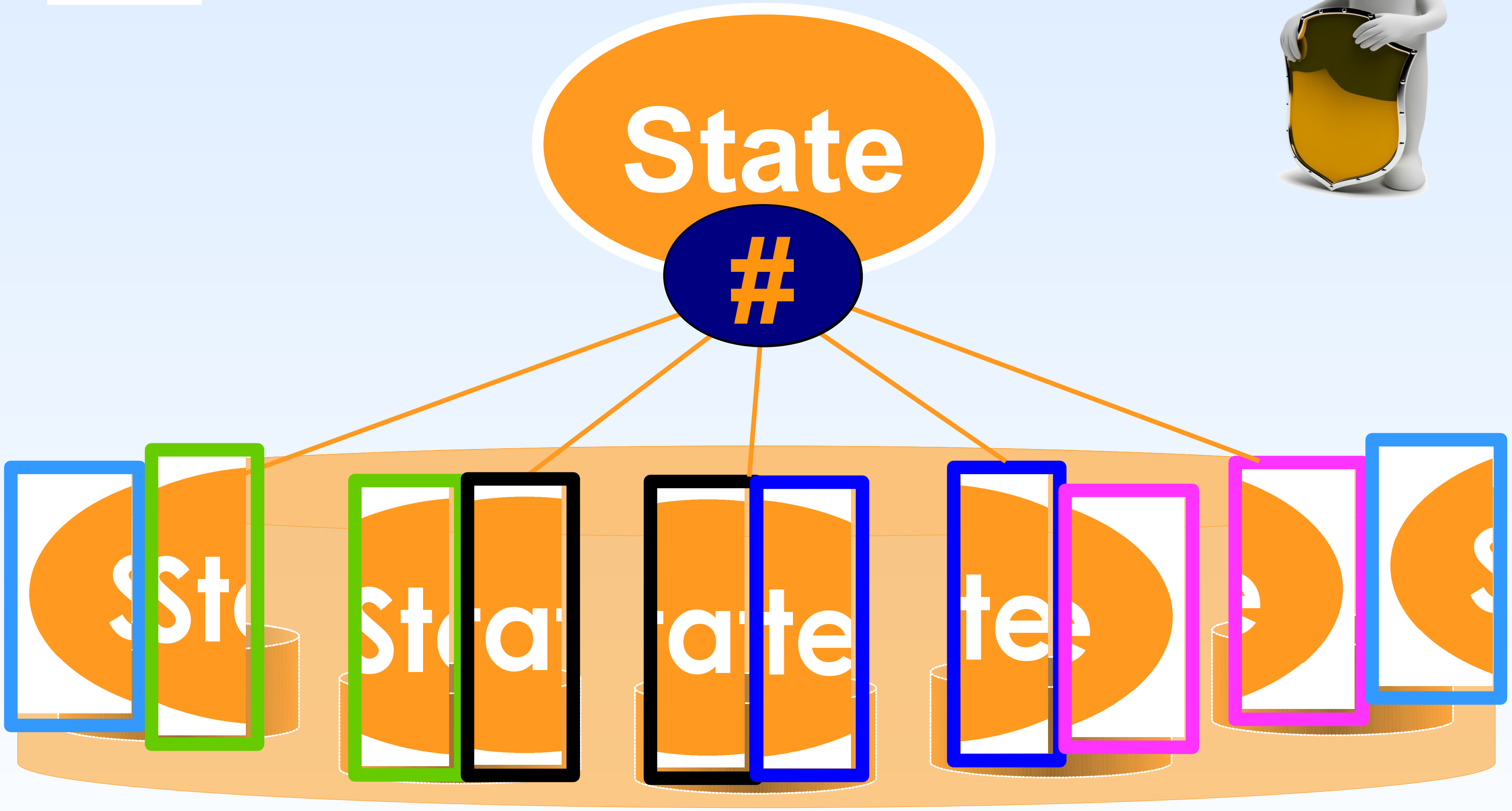
- Horizontal Partitioning
- Snowflakes & Clones
- Single-Thread Execution
- Compiled-In Queries

Partitions



VoltDB slices data tables horizontally using a hash over the most significant primary key.

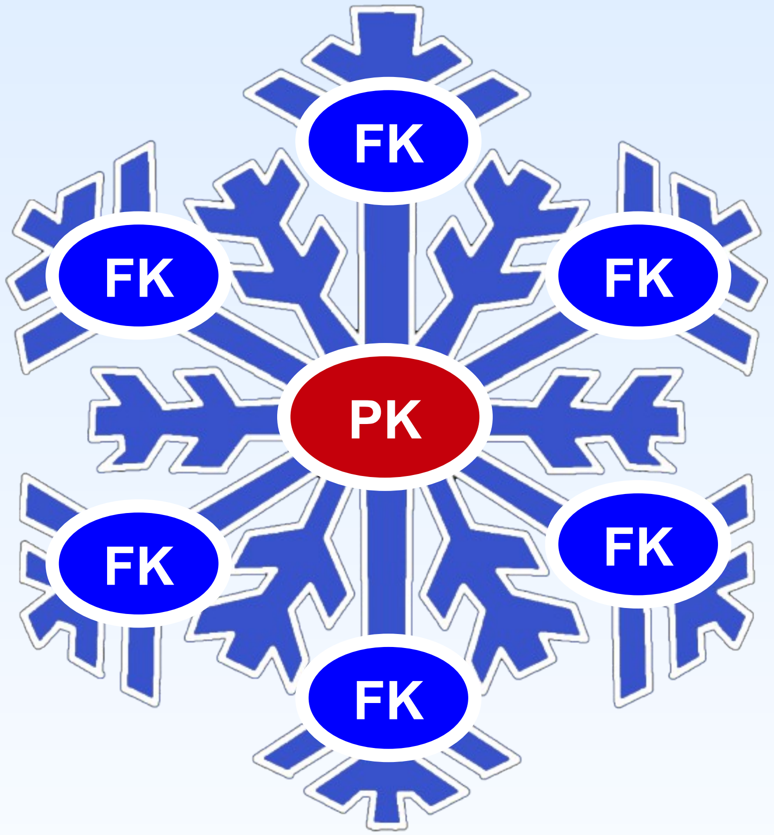
K-Safety



VoltDB replicates within the cluster, across servers. The for-pay edition can replicate to other data centers.

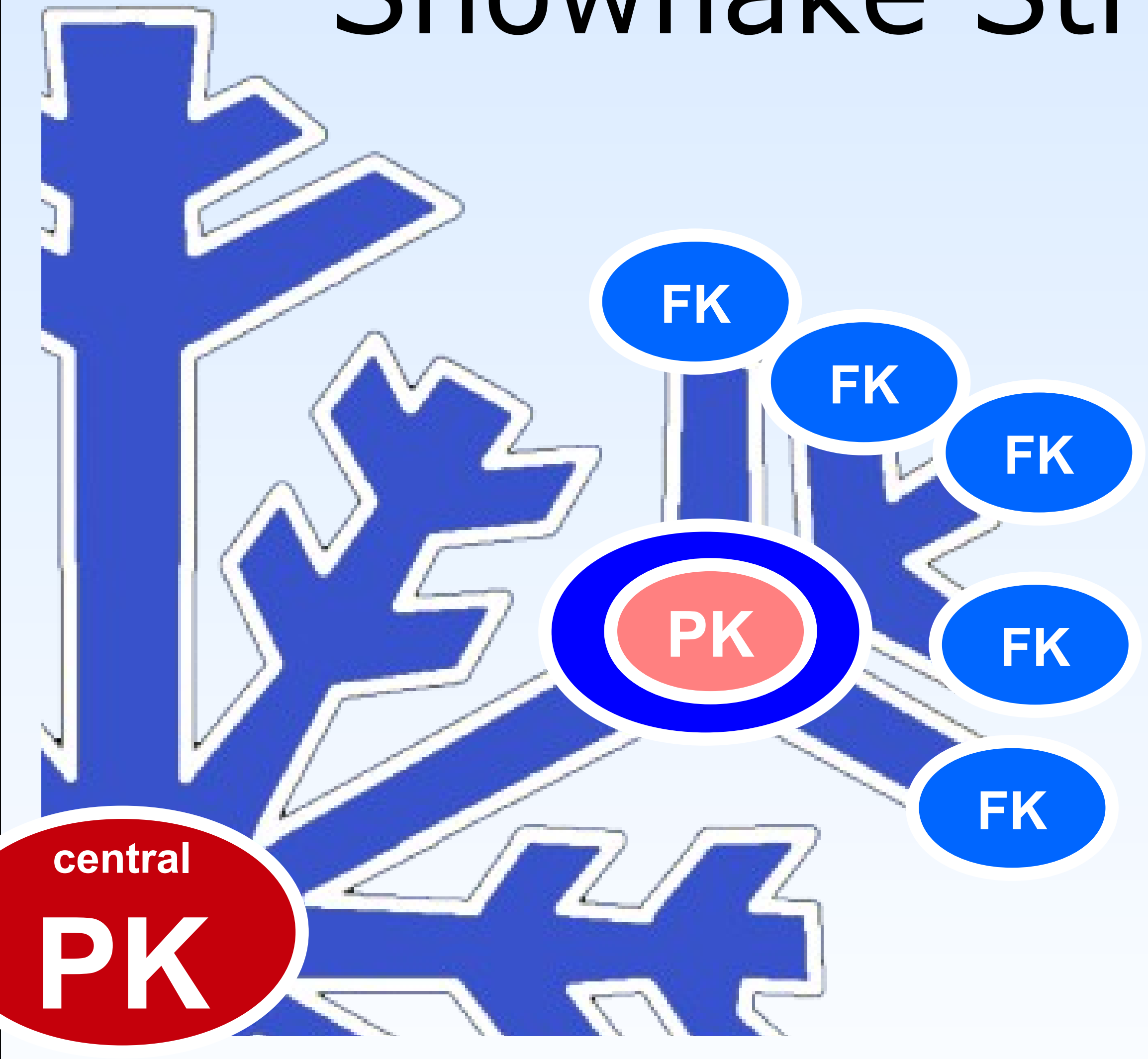
VOLTDB

Snowflake Structures



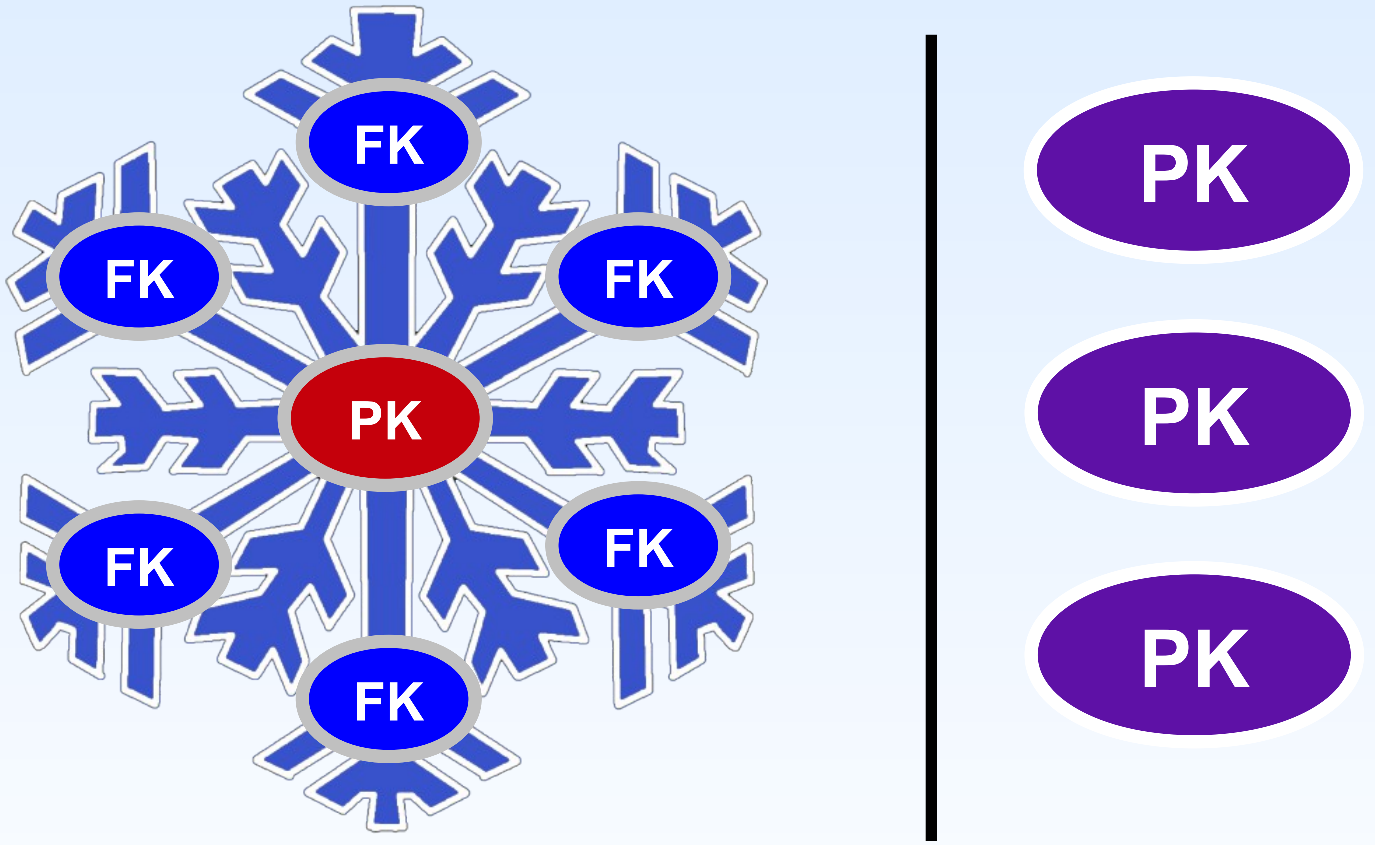
VoltDB clustering uses the fact that most real world data is shaped like a 'snowflake'.

Snowflake Structures



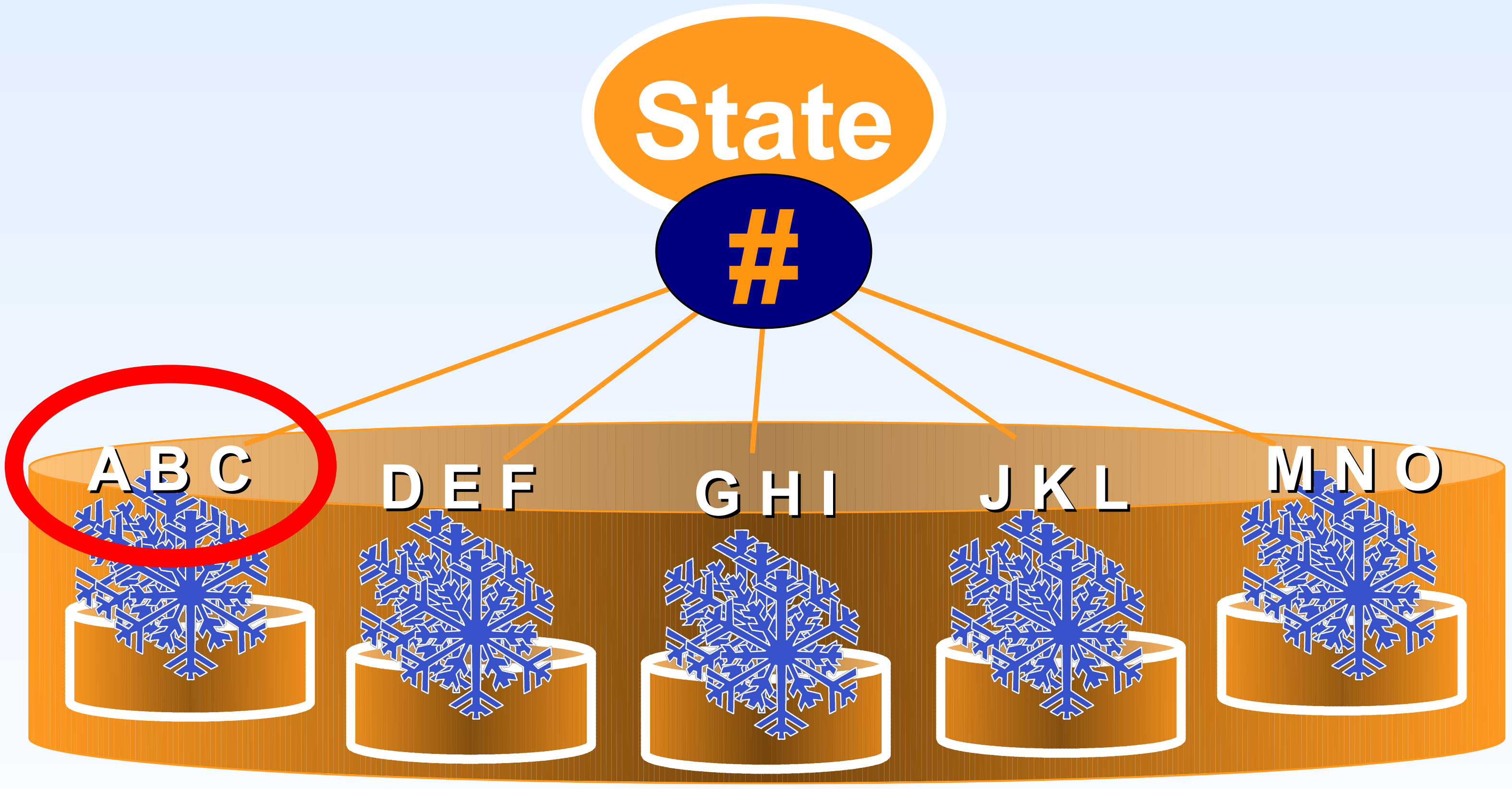
A snowflake can have sub branches. All point to one main, central primary key.

Non-Aligned Data



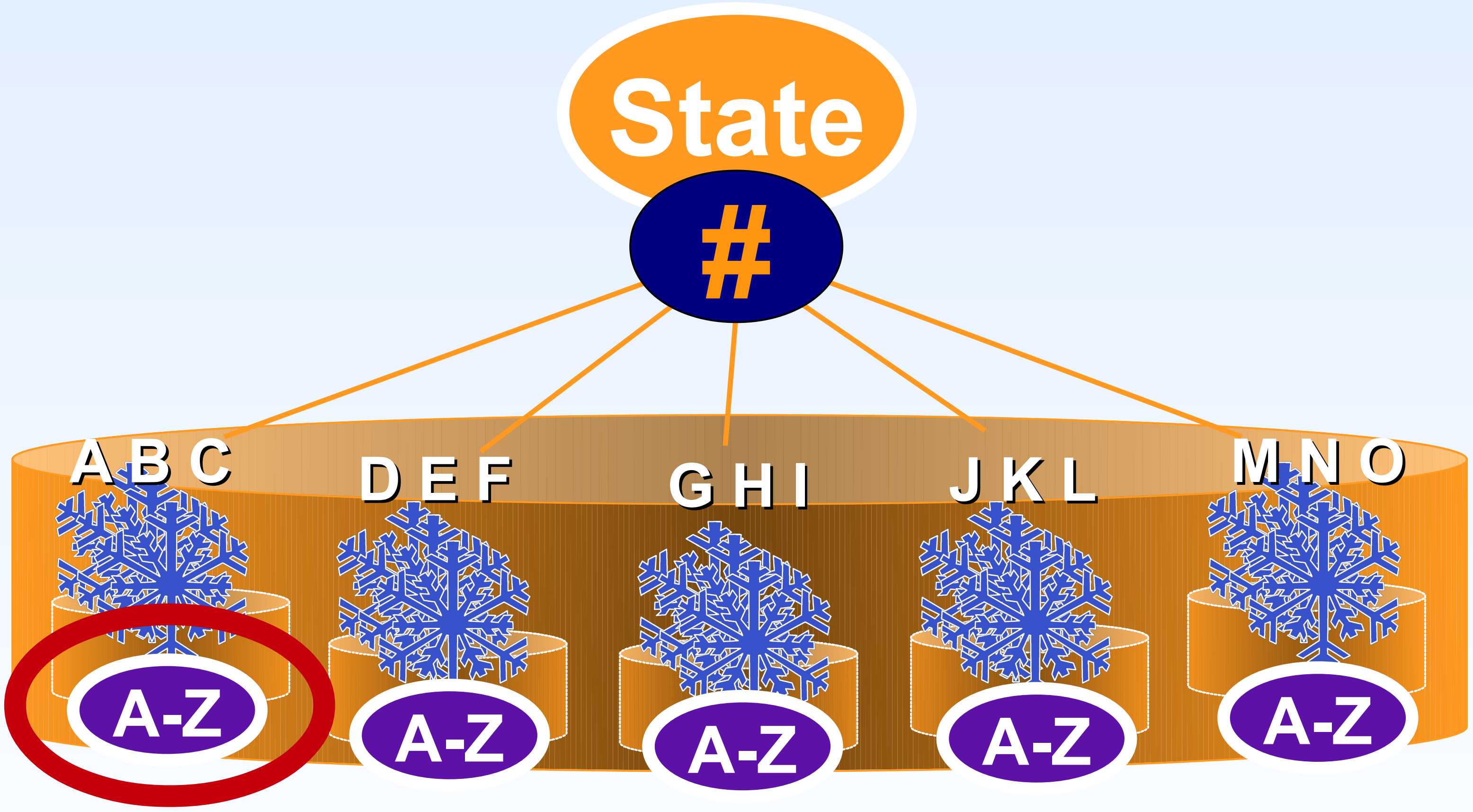
Some data will **not** fit into the snowflake. And that's fine. VoltDB does **not** partition it but **replicates** it on every node.

Snowflakes are Partitioned



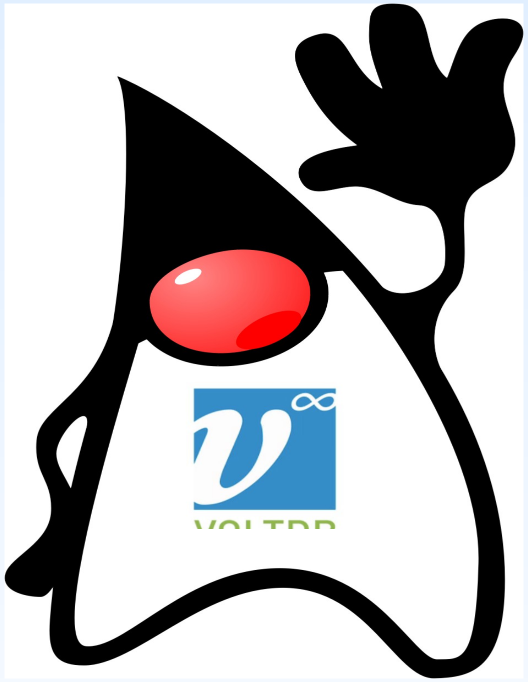
Partitioning is facilitated by a hash over the primary key of the snowflakes.

NON-Flakes are Replicated



The special 'non-snowflake replication' is transparent for the user and allows for single-threaded execution.

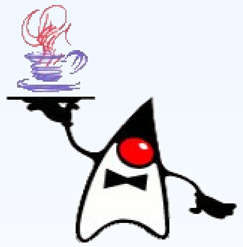
Stored Procedures



Stored procedures in VoltDB must be wrapped in Java.

Stored Procedures

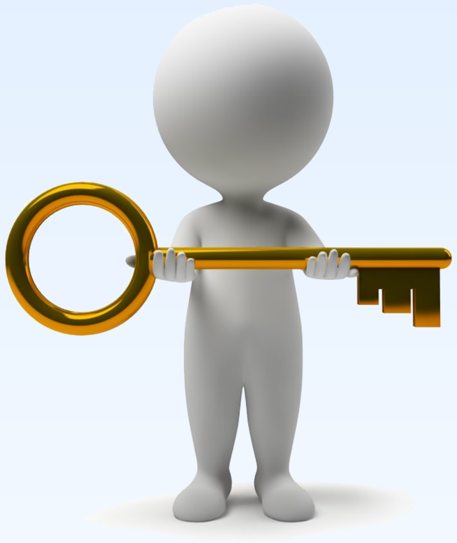
```
import org.voltdb.*;

public class Select extends VoltProcedure {

  public final SQLStmt sql = new SQLStmt(
      "SELECT HELLO, WORLD FROM HELLOWORLD WHERE DIALECT = ?;"
  );

  public VoltTable[] run( String language)
      throws VoltAbortException {
          voltQueueSQL( sql, language );
          return voltExecuteSQL();
      }
}
```



This is the complete Java source and SQL of the sample class called “Select” that we”ll meet again later.

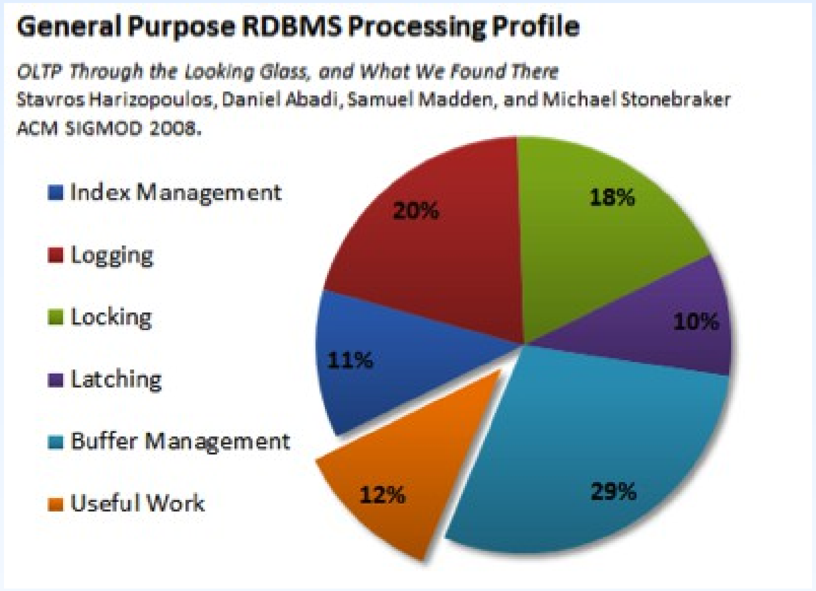
Single Threaded

- One Thread per Partition =
- One Thread per Transaction
- One Thread can't race itself



Lockless

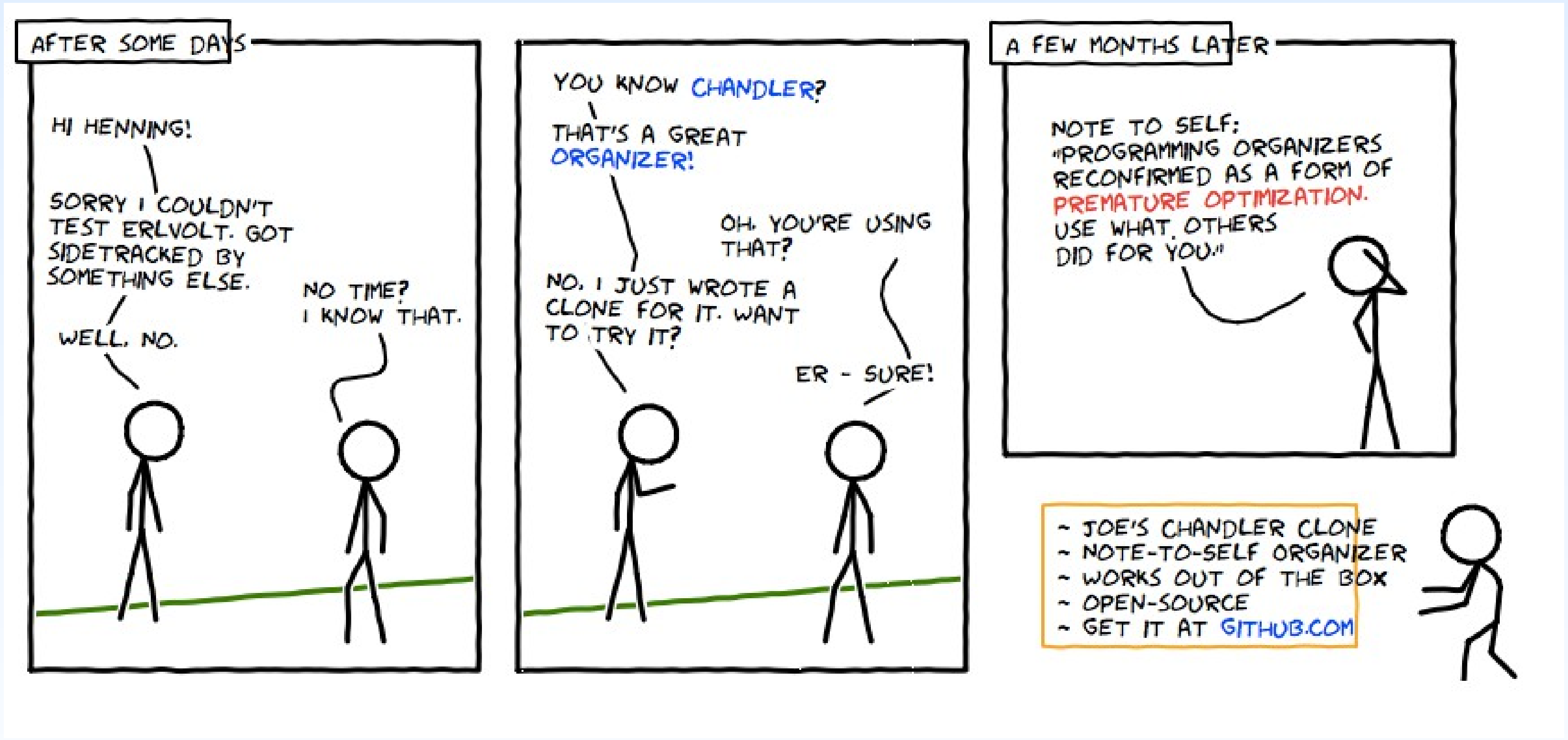
"Transact to Completion"


General Purpose RDBMS Processing Profile

OLTP Through the Looking Glass, and What We Found There
Stavros Harizopoulos, Daniel Abadi, Samuel Madden, and Michael Stonebraker
ACM SIGMOD 2008.

- Uses all cores
- Sheds 75% of DBM work load
- Speed up of two magnitudes

Task Scheduling


AFTER SOME DAYS
HI HENNING!
SORRY I COULDN'T TEST ERLVOLT. GOT SIDETRACKED BY SOMETHING ELSE.
WELL, NO.
NO TIME? I KNOW THAT.
YOU KNOW CHANDLER?
THAT'S A GREAT ORGANIZER!
OH, YOU'RE USING THAT?
NO, I JUST WROTE A CLONE FOR IT. WANT TO TRY IT?
ER - SURE!
A FEW MONTHS LATER
NOTE TO SELF: "PROGRAMMING ORGANIZERS RECONFIRMED AS A FORM OF PREMATURE OPTIMIZATION. USE WHAT OTHERS DID FOR YOU."
- JOE'S CHANDLER CLONE
- NOTE-TO-SELF ORGANIZER
- WORKS OUT OF THE BOX
- OPEN-SOURCE
- GET IT AT GITHUB.COM

Joe Armstrong invites you to look at his Erlang version of Chandler.

Using it with Erlang

Erlvolt

Insert

```
erlvolt:call_procedure(Connection, "Insert", ["안녕하세요", "세계", "Korean"]),
```

Select

```
Response = erlvolt:call_procedure(Connection, "Select", ["Korean"]),

Row = erlvolt:fetch_row(Table, 1),

io:format("~n~n~s, ~s!~n",
[ erlvolt:get_string(Row, Table, "HELLO"),
  erlvolt:get_string(Row, Table, "WORLD") ]);
```

Erlvolt is a native Erlang driver for VoltDB. It is as fast as the drivers for VoltDB for other languages that use C.

Hello Erlvolt

```
-module(hello).
-export([run/0]).
-include("erlvolt.hrl").

run() ->

    crypto:start(),
    application:start(erlvolt),

    erlvolt:add_pool(hello_pool, [{"localhost", 21212}]),

    Result = erlvolt:call_procedure(hello_pool, "Select", ["Swedish"]),

    Table = erlvolt:get_table(Result, 1),
    Row = erlvolt:get_row(Table, 1),
    Hello = erlvolt:get_string(Row, Table, "HELLO"),

    io:format("~n~s World!~n~n", [Hello]),

    erlvolt:close_pool(hello_pool).
```

This is a complete hello world example, opening the database connection, querying and fetching results.

Examples

```
erlvolt$ ls examples

Makefile          hello_pre3.erl        voter.erl
hello.erl         parallel.erl
hello_plus.erl    parallel_pre3.erl
```

This is a complete hello world example, opening the database connection, querying and fetching results.

Bench

```
erlvolt$ ls etc/bench

Makefile       README.md     bench.erl     benchstart
README.html    bench.config  bench.totals
```

This is a complete hello world example, opening the database connection, querying and fetching results.

The Benchmark

Benchmark

"TV Contest Voters" Sample

- Millions of callers
- Small set of candidates
- Massive peak
- One transaction is one vote
- Callers are identified by their phone number
- Callers must not be allowed to vote twice

The "Voter" sample is a VoltDB staple, honestly demonstrating its strength of high peaks and fast answers.

Benchmark

- Amazon EC2 CC2 cluster instances
- 128 core client clusters + 192 core VoltDB cluster
- 877,519 transactions per second (TPS)
- 3,510,076 operations per second
- 260,000 TPS per 16 core client
- 26,500 transactions/second per CPU core
- Stable, also under overload
- Pretty much linear scale

Details: http://blog.voltdb.com/877000-tps-with-erlang-and-voltdb/

The number of ~25,000 transactions per core seems to be the most valuable result to base predictions on.

Benchmark DDL

```
CREATE TABLE contestants
(
    contestant_number integer     NOT NULL,
    contestant_name   varchar(50) NOT NULL,
    CONSTRAINT PK_contestants PRIMARY KEY (contestant_number)
);

CREATE TABLE votes
(
    phone_number       bigint     NOT NULL,
    state              varchar(2) NOT NULL,
    contestant_number  integer    NOT NULL
);

CREATE TABLE area_code_state
(
    area_code smallint   NOT NULL
    state     varchar(2) NOT NULL
    CONSTRAINT PK_area_code_state PRIMARY KEY (area_code)
);
```

Benchmark DQL

Each Transaction has 4 Operations

```
// Check if the vote is for a valid contestant
SELECT contestant_number FROM contestants WHERE contestant_number = ?;

// Check if the voter has exceeded their allowed number of votes
SELECT num_votes FROM v_votes_by_phone_number WHERE phone_number = ?;

// Check an area code to retrieve the corresponding state
SELECT state FROM area_code_state WHERE area_code = ?;

// Record a vote
INSERT INTO votes (phone_number, state, contestant_number) VALUES (?, ?, ?);
```

Each transaction performs three searches across the entire data set, and a write. Additionally, 2 matierialized views are updated.

Volt's SQL

SQL Statements

Supported SQL statements

- DELETE
- INSERT
- SELECT
- UPDATE

Supported standard SQL DDL statements

- CREATE INDEX
- CREATE TABLE
- CREATE VIEW

SQL Functions

Column Aggregation Functions

• AVG() • COUNT() • MAX() • MIN() • SUM()

Date Function

• EXTRACT()

JSON Functions

• ARRAY_ELEMENT() • ARRAY_LENGTH() • FIELD()

Logic and Conversion Functions

• CAST() • DECODE()

Math Function

• ABS() • CEILING() • EXP() • FLOOR() • POWER() • SQRT()

String Functions

• CHAR_LENGTH() • CONCAT() • LEFT() • OCTET_LENGTH()
• POSITION() • REPEAT() • RIGHT() • SPACE() • SUBSTRING()

SQL Extensions

VoltDB-specific extensions for Stored Procedures

- CREATE PROCEDURE AS
- CREATE PROCEDURE FROM CLASS
- CREATE ROLE

VoltDB-specific extensions for Partitioning

- PARTITION PROCEDURE
- PARTITION TABLE
- EXPORT TABLE

Wrapping Up

Resources

Erlvolt

Dowload
https://github.com/Eonblast/Erlvolt

Benchmark
http://blog.voltdb.com/877000-tps-with-erlang-and-voltdb/

Installation
https://gist.github.com/hdiedrich/5415065

VoltDB

Download
http://voltdb.com/products-services/downloads

Webinars
http://community.voltdb.com/weninars

Forum
http://community.voltdb.com/forum

The Voter Example
https://github.com/VoltDB/voltdb/tree/master/examples/voter

877k Benchmark Blog Post
http://voltdb.com/company/blog/695k-tps-nodejs-and-voltdb

On Volt's Origins
http://nms.csail.mit.edu/~stavros/pubs/hstore.pdf

SumUp

http://www.sumup.com

Eonblast

http://www.eonblast.com

Questions

- Email: hdiedrich@eonblast.com
- Twitter: @hdiedrich
- Forum: http://community.voltdb.com/forum

Try Erlvolt

Install & run a local benchmark:

```
git clone -b voltdb-3.7.0.5 git://github.com/VoltDB/voltdb.git
git clone git://github.com/Eonblast/Erlvolt.git erlvolt
cd voltdb && ant && cd examples/voter && ./run.sh &
cd erlvolt && make profile bench
```

Trying it yourself takes only minutes including the full build. More details here: https://gist.github.com/hdiedrich/5415065