VoltDB

VOLTDB

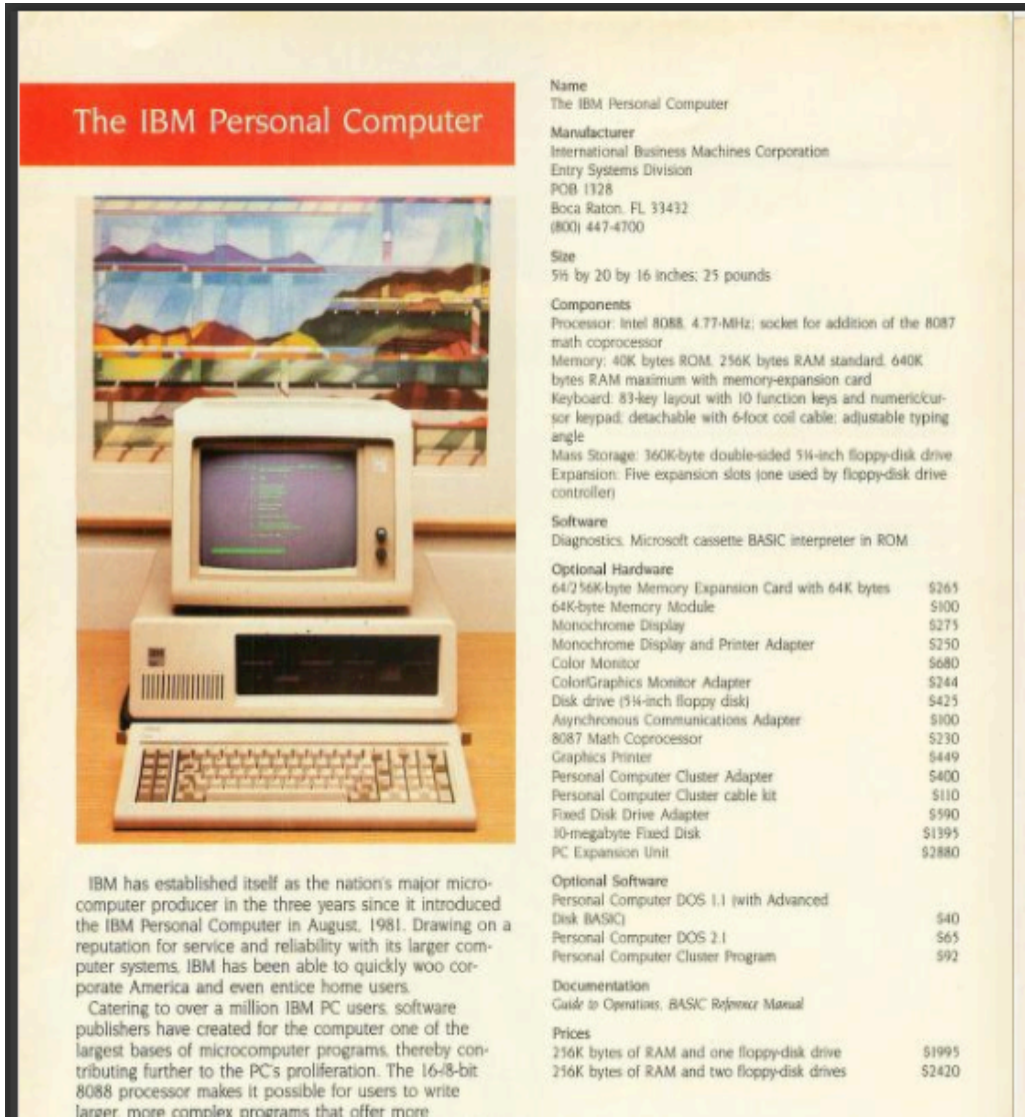
Intelligent Ingestion

David Rolfe
Director of Solution Architecture, EMEA

17-Jul-18

In 1984 the world was a very different place…

- Ronald Reagan was president
- For only US$3700 (about US$8700 now) you could buy an IBM PC with 256KB of RAM and a 1MB hard drive.
- Multi CPU computers were an exotic rarity.
- There was almost no computer to computer connectivity. Computers interacted with humans. Slowly.
- 1984 is roughly when the architecture of the major RDBMS products was designed.



The IBM Personal Computer

IBM has established itself as the nation's major microcomputer producer in the three years since it introduced the IBM Personal Computer in August, 1981. Drawing on a reputation for service and reliability with its larger computer systems, IBM has been able to quickly woo corporate America and even entice home users.

Catering to over a million IBM PC users, software publishers have created for the computer one of the largest bases of microcomputer programs, thereby contributing further to the PC's proliferation. The 16-/8-bit 8088 processor makes it possible for users to write larger, more complex programs that offer more

Name
The IBM Personal Computer

Manufacturer
International Business Machines Corporation
Entry Systems Division
POB 1328
Boca Raton, FL 33432
(800) 447-4700

Size
5½ by 20 by 16 inches; 25 pounds

Components
Processor: Intel 8088, 4.77-MHz; socket for addition of the 8087 math coprocessor
Memory: 40K bytes ROM, 256K bytes RAM standard, 640K bytes RAM maximum with memory-expansion card
Keyboard: 83-key layout with 10 function keys and numeric/cursor keypad; detachable with 6-foot coil cable; adjustable typing angle
Mass Storage: 360K-byte double-sided 5¼-inch floppy-disk drive
Expansion: Five expansion slots (one used by floppy-disk drive controller)

Software
Diagnostics, Microsoft cassette BASIC interpreter in ROM

Optional Hardware

64/256K-byte Memory Expansion Card with 64K bytes	$265
64K-byte Memory Module	$100
Monochrome Display	$275
Monochrome Display and Printer Adapter	$250
Color Monitor	$680
Color/Graphics Monitor Adapter	$244
Disk drive (5¼-inch floppy disk)	$425
Asynchronous Communications Adapter	$100
8087 Math Coprocessor	$230
Graphics Printer	$449
Personal Computer Cluster Adapter	$400
Personal Computer Cluster cable kit	$110
Fixed Disk Drive Adapter	$590
10-megabyte Fixed Disk	$1395
PC Expansion Unit	$2880

Optional Software

Personal Computer DOS 1.1 (with Advanced Disk BASIC)	$40
Personal Computer DOS 2.1	$65
Personal Computer Cluster Program	$92

Documentation
Guide to Operations, BASIC Reference Manual

Prices

256K bytes of RAM and one floppy-disk drive	$1995
256K bytes of RAM and two floppy-disk drives	$2420

Then we had the era of the "One Big Database"…

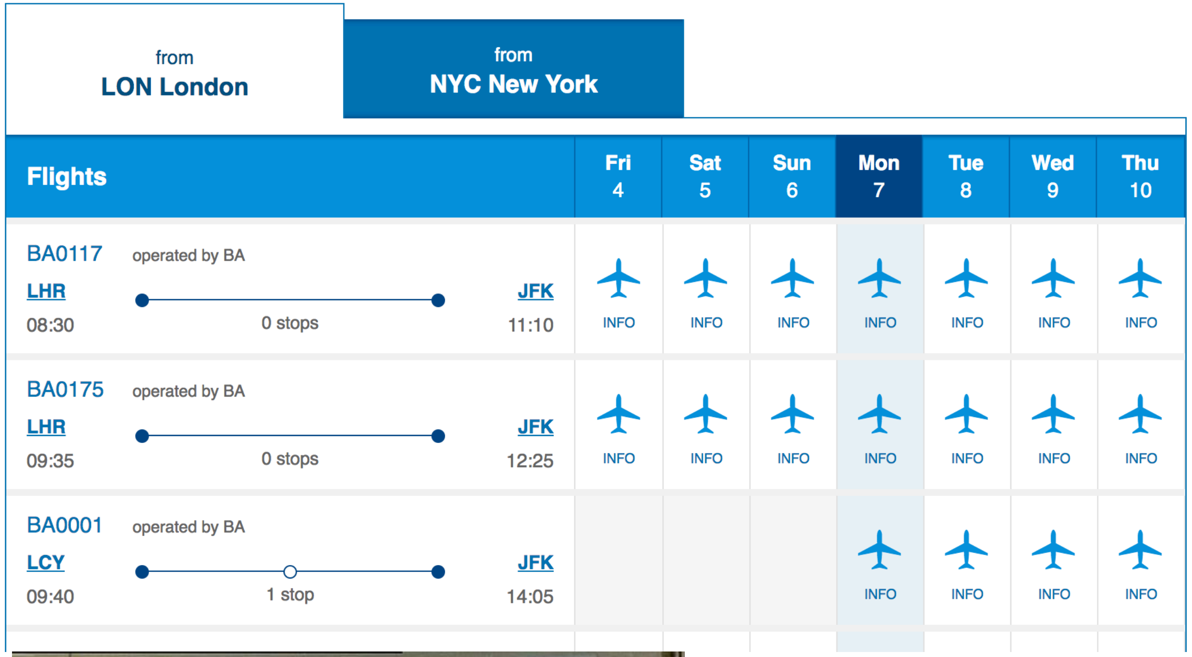
- Organizations realized that data was *valuable.*
- Very bad practice for Use Cases to dictate data structures
  - Optimization regarded as a last resort. And a sign of failure.
- The database was a repository of corporate data.
- There was one centrally planned database schema that was 'correct'.
- Everyone was supposed to use it. Or else.
- *With hindsight it was all a bit "Soviet"…*

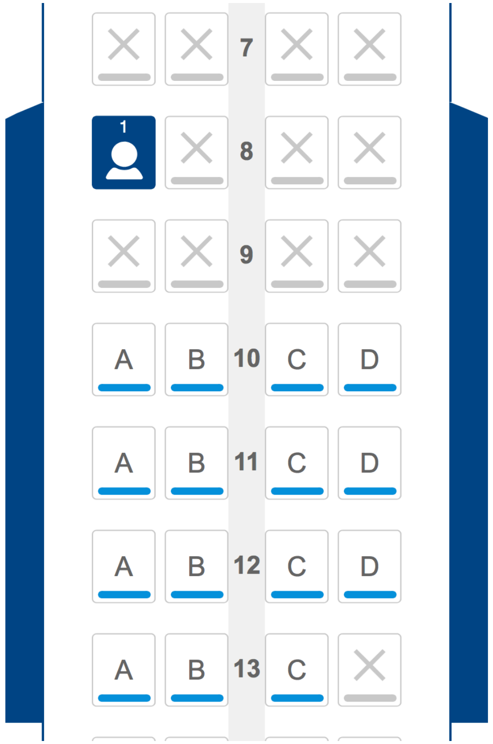
It may have been ”Soviet” but…

- Databases were a new and radical concept that solved a real problem.
- They were inspired by the industrial scale chaos caused by every application ‘owning’ its own data
  - Some of that chaos has returned
- Regarding data as an asset that is more important than a single Use Case is ‘common sense’ now.
- The “One Big Database” was killed by:
  - Office politics/organizational complexity
  - Operational and design complexity.
  - Lack of suitably miraculous products.
  - The PC and virtualization.
- We now have gone to the other extreme – microservices and KV stores

Modern applications are incredibly demanding

Goal: Predict flight delays.





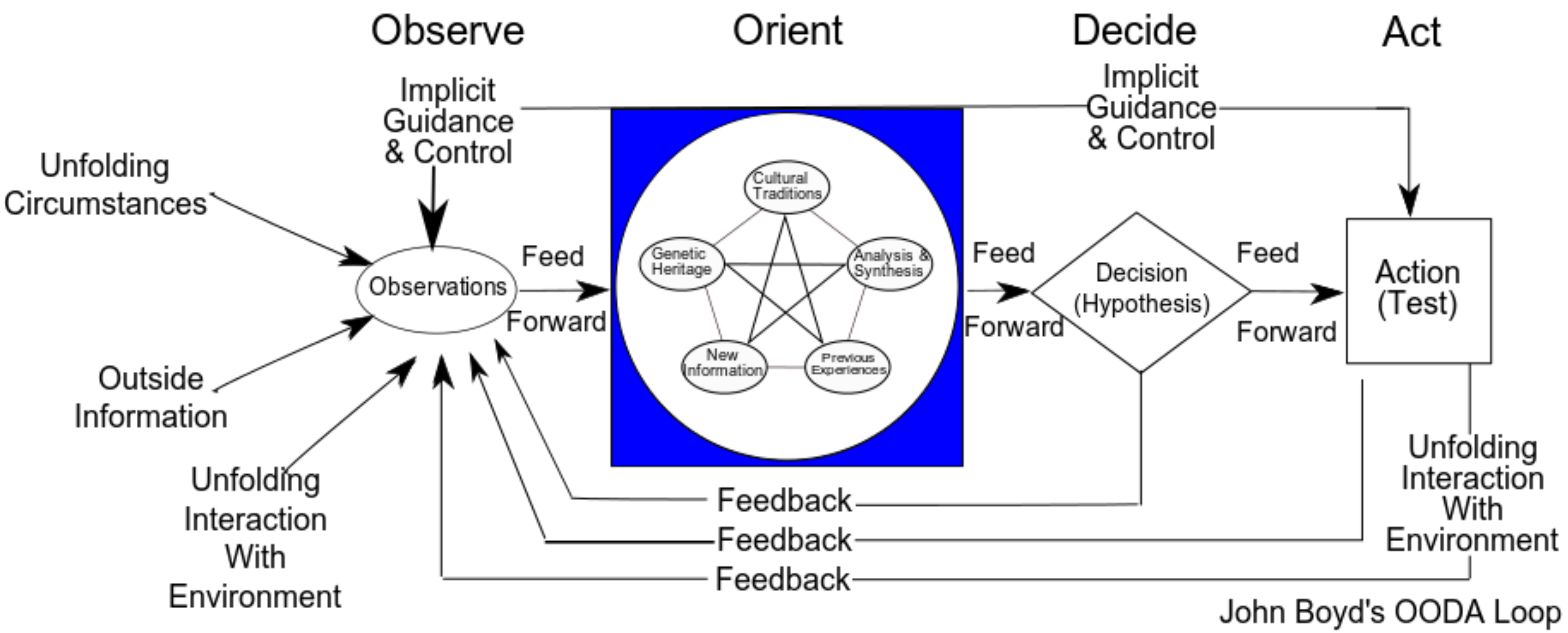
"The Late Arrival Of The Incoming Aircraft"

Raw TAF
KJFK 070809Z 0708/0812 36004KT P6SM SCT025 BKN040
FM071400 04009KT P6SM SCT035 BKN050
FM071800 15010G15KT P6SM SCT035 BKN050
FM080100 09009KT P6SM SCT030 BKN100
FM080900 05005KT P6SM SCT020 SCT100

Raw METAR
KJFK 070951Z 35006KT 10SM FEW060 BKN250 13/11 A3000 RMK AO2 SLP159 T01280106
KJFK 070851Z 35005KT 10SM FEW060 BKN250 12/11 A2998 RMK AO2 SLP152 T01220106 53013
KJFK 070751Z 36004KT 10SM FEW055 BKN250 13/11 A2996 RMK AO2 SLP146 T01330106
KJFK 070651Z 36008KT 10SM SCT024 BKN055 14/11 A2996 RMK AO2 SLP144 T01440111

VOLTDB

The OODA Loop, and why it's important…



John Boyd's OODA Loop


John "Forty Second" Boyd

1. In order to 'win' your application's loop needs to run faster than 'reality'
2. 'Winning' against a web page means a loop of under 7 seconds.
3. 'Winning' in an IoT context means a loop of around 7 milliseconds.
4. 'Winning' now involves complex decision making at millisecond timescales.

VOLTDB

Complexity, Latency and Volumes

Year	Internet Bandwidth (GB)	Required Response Time)	Application Complexity
1985	33	2 day batch turnaround	Row level locking being invented. Transactions of any sort rare.
1995	150,500	2 minute batch	Row level locking of 5-10 rows; web servers using databases.
2000	75,250,000	8000ms – Web Page Advertising	Web advertising: Cookie -> User -> Demographic -> Ad
2010	19,974,008,812	100ms - Video Game Analytics / Sophisticated web advertising	Decisions need to consider dozens/hundreds of elements
2015-2018	>42,423,169,029	10ms - IoT / Many Devices	Decisions may need to consider thousands of elements

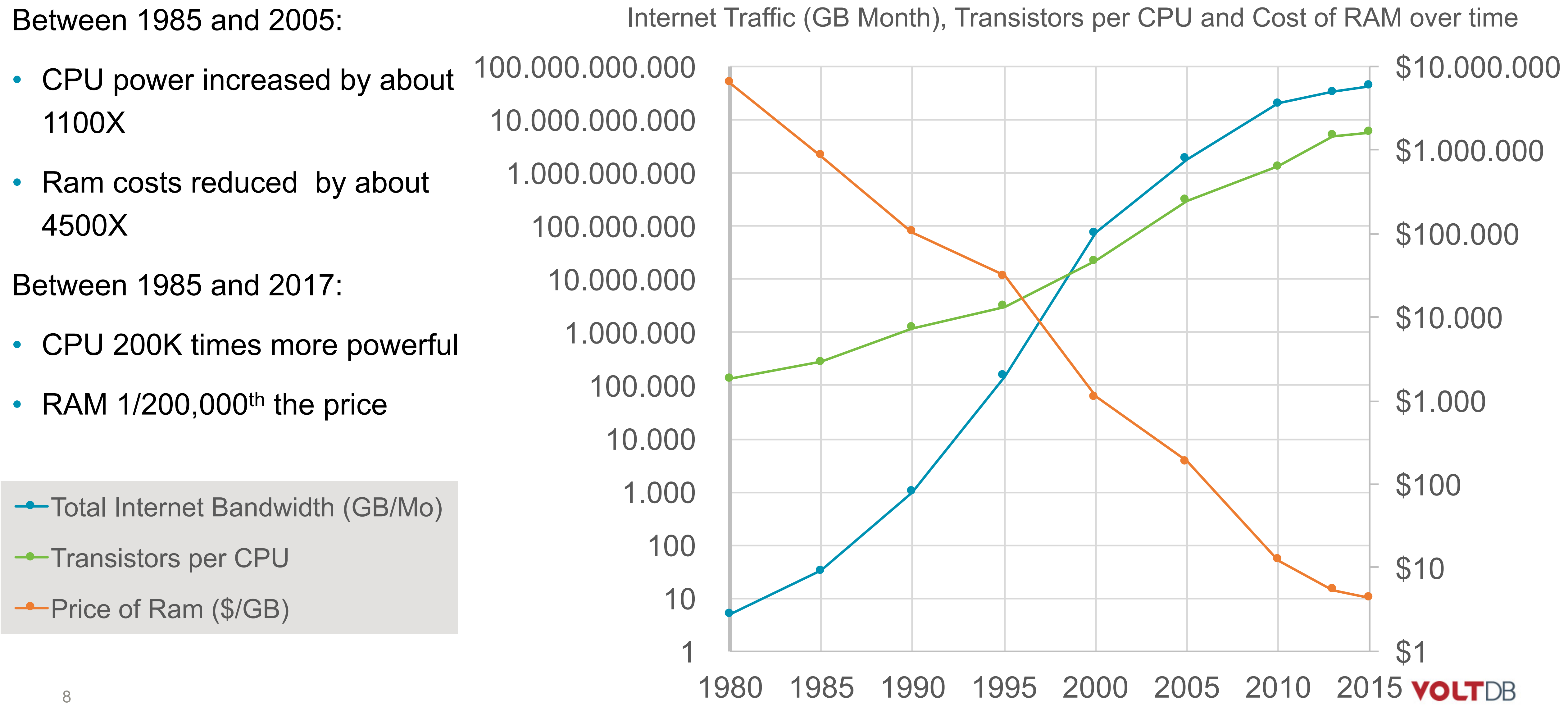
Between 1984 and 2015 a lot changed...

Between 1985 and 2005:

- CPU power increased by about 1100X
- Ram costs reduced by about 4500X

Between 1985 and 2017:

- CPU 200K times more powerful
- RAM 1/200,000th the price



VOLTDB

We now face real ‘pain’…

- Timescales (“OODA loop”), volumes and complexity all pose challenges.

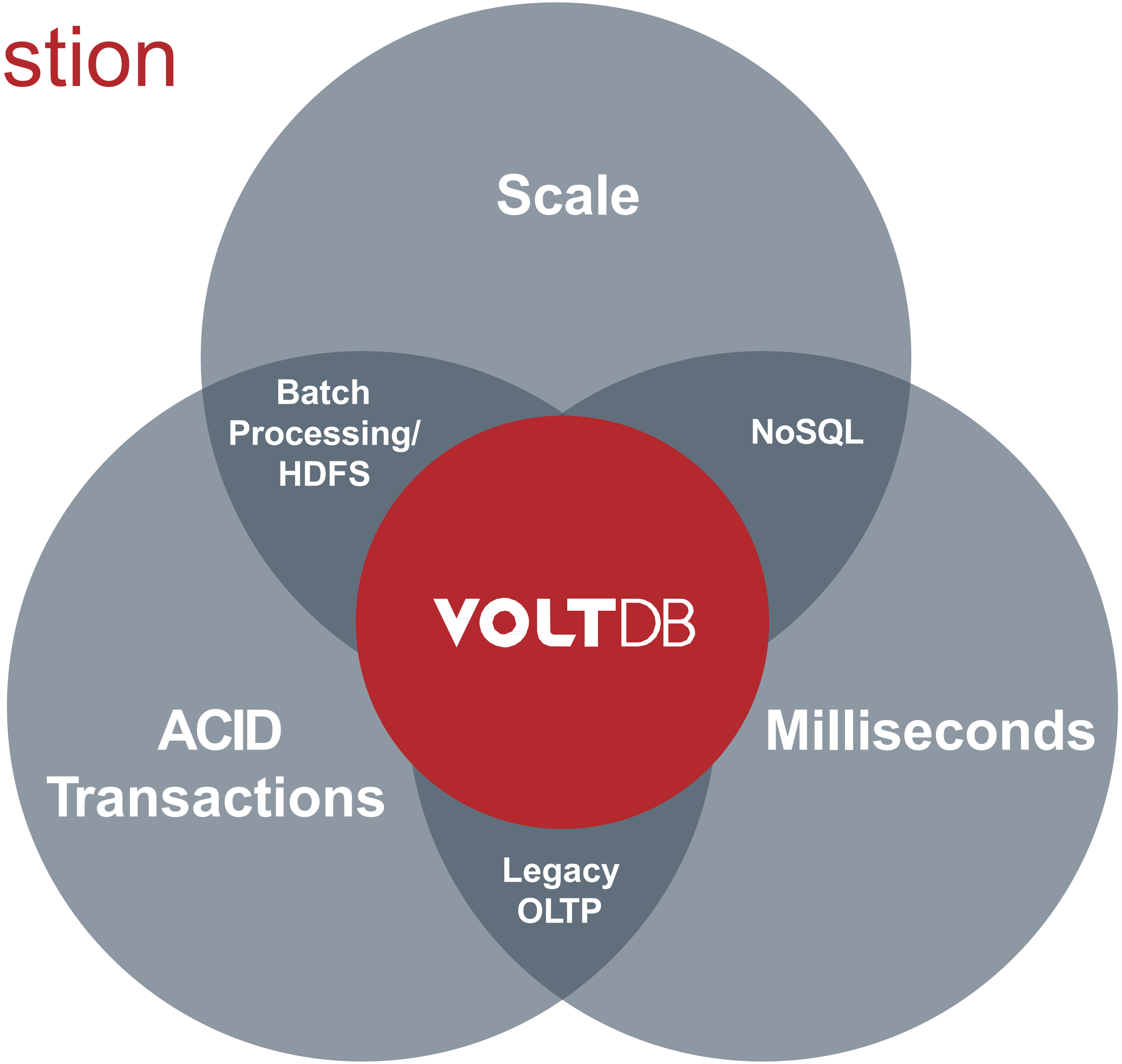
- We need to take complex decisions, fast:

  - Complex Decisions usually involve many different data sources.
  - Key Value solutions imply a large number of network trips…
  - …and ACID becomes an issue if your data changes while you are reading it
  - Legacy RDBMS implies a lack of scale.
  - Decisions involving shared resources usually implies locking or retries
  - Many decisions involve aggregate values (”HTAP/Translytics”)
  - Many decisions involve telling a third party something

- What we need is the ability to do all of this *as the data arrives*.

Intelligent Ingestion

1. Complex Decisions
2. Multiple Data Sources
3. ACID
4. Millisecond Timing
5. Massive Scale



Why use VoltDB for “Intelligent Ingestion”?

Intelligent Ingestion

VoltDB is optimized for Scale, ACID and Latency…

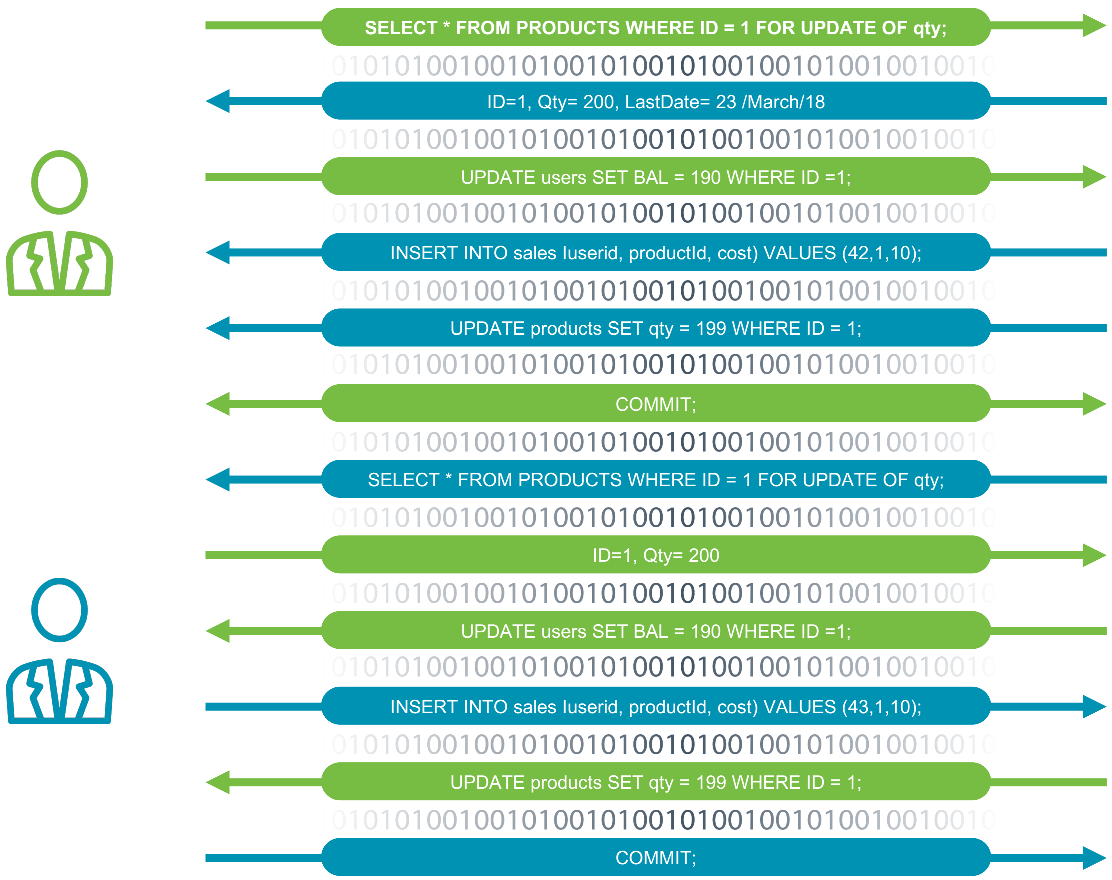
An **Ariel Atom**. Very good for going round racetracks very quickly in nice weather. Not so good for school runs, driving tests, shopping, off road activities, as an ambulance, polar exploration, amphibious assaults, carrying cargo….

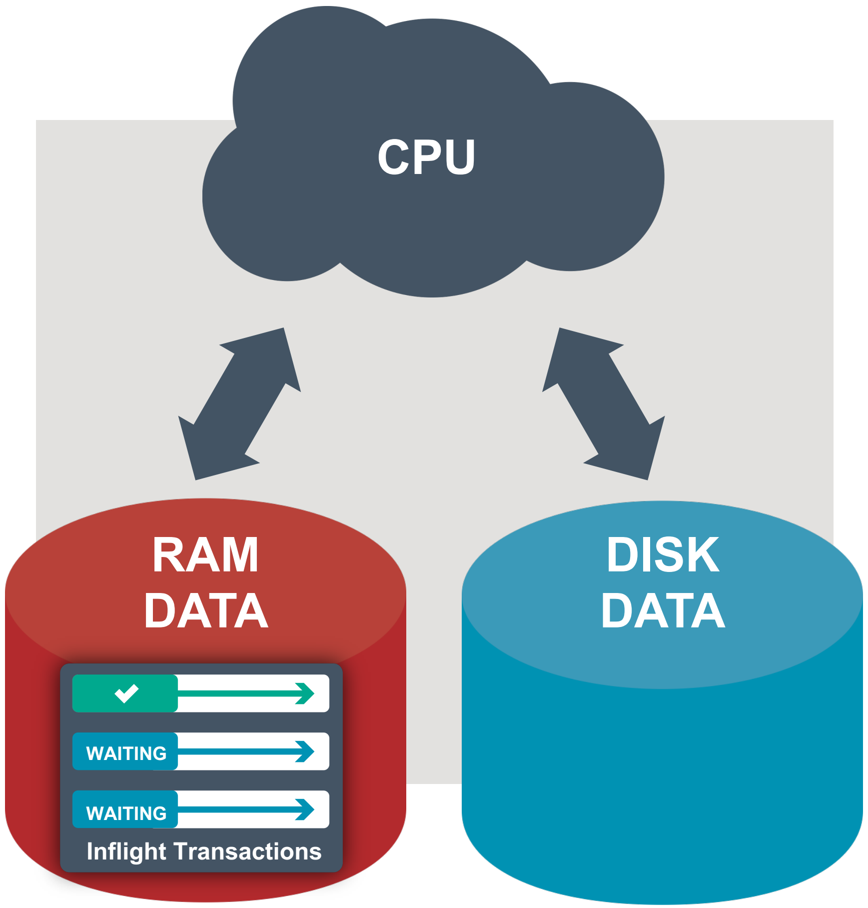
VoltDB is optimized to solve one class of problems better than anything else that exists.

It is not a general purpose RDBMS, nor a replacement for one.

It's used to complement your existing stack rather than replacing it.

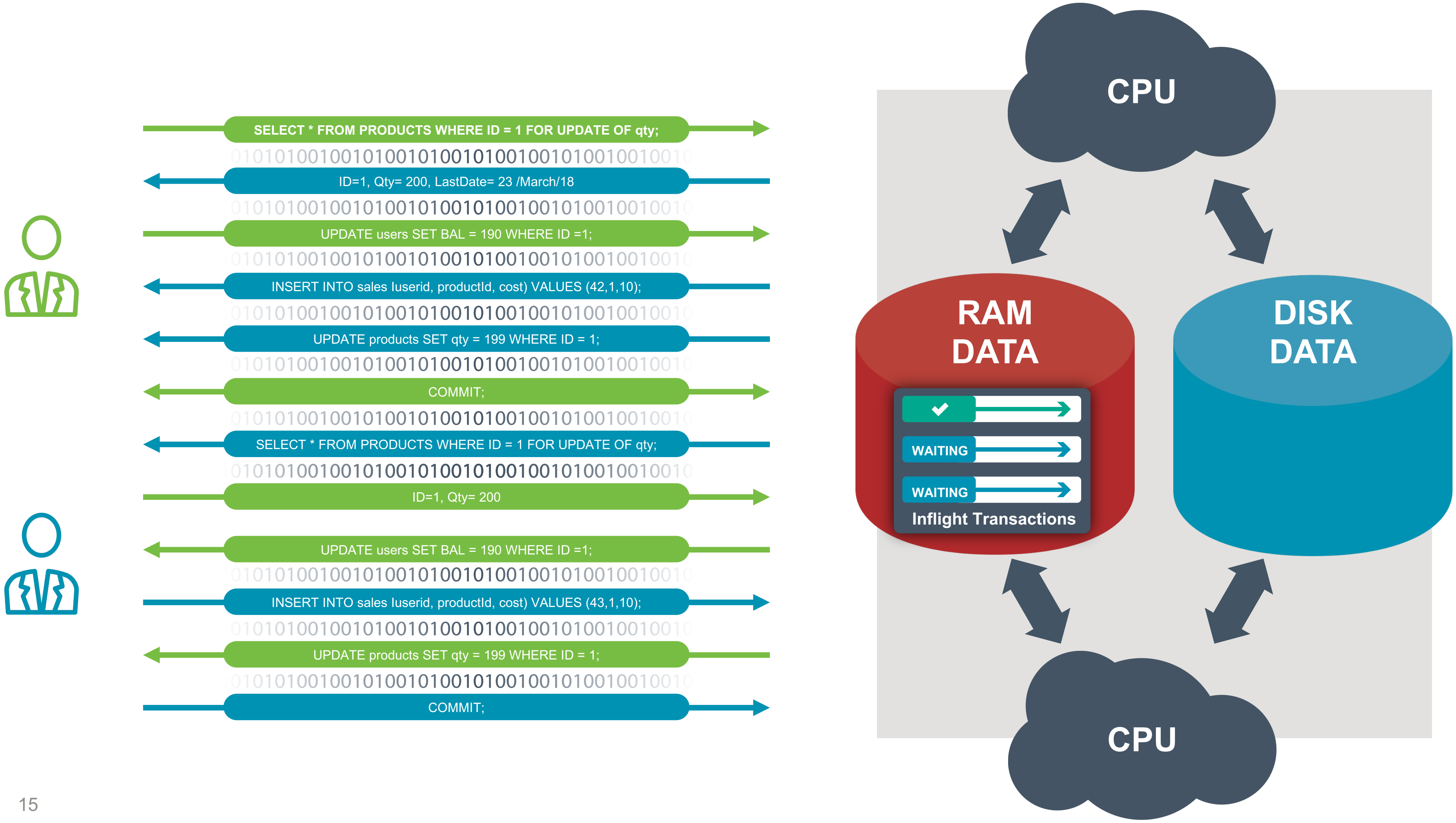
RDBMS - How We Thought an RDBMS Worked



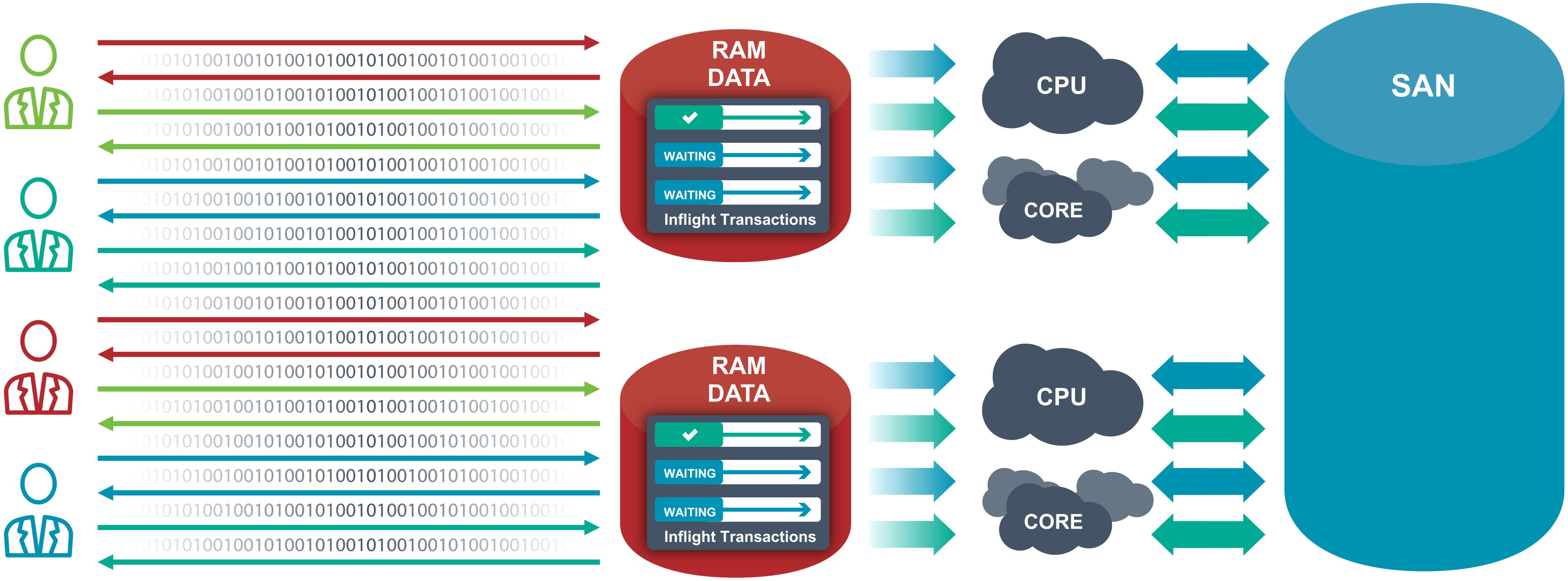


VOLTDB

RDBMS - What Actually Happens – Part 1…

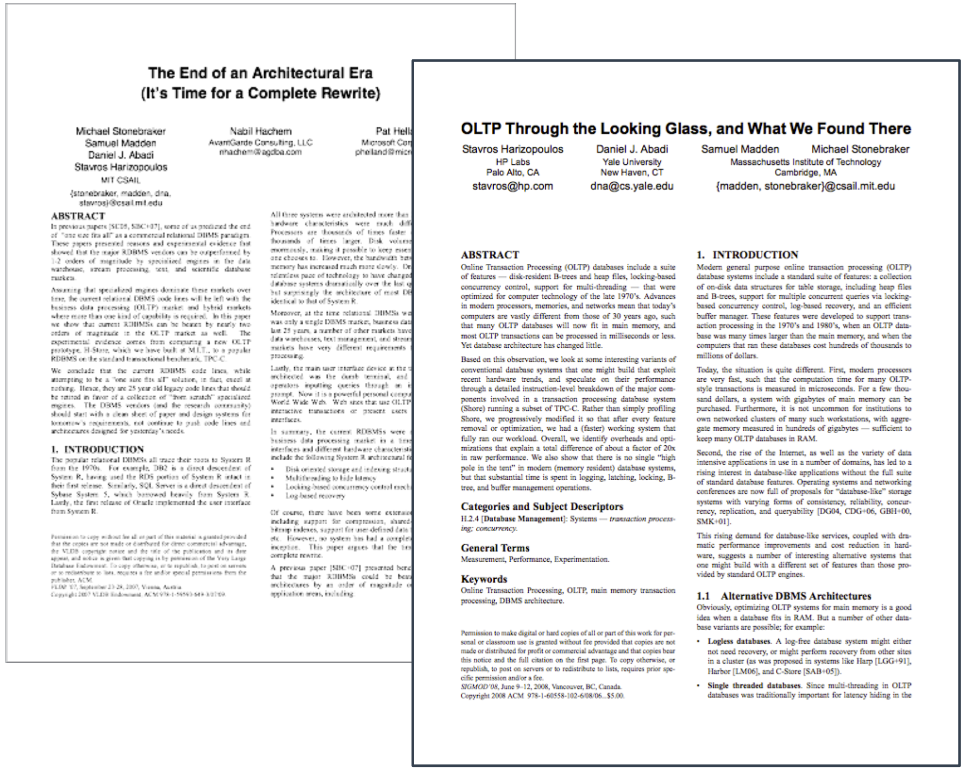


RDBMS - What Actually Happens – Part 2



VOLTDB

VoltDB was designed to solve this problem



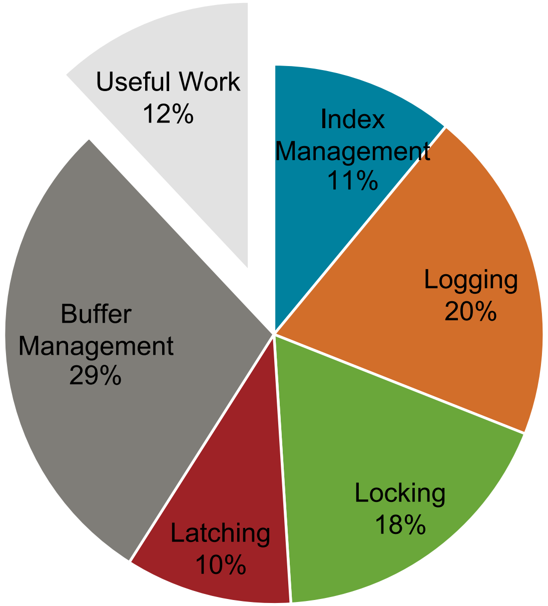
The End of an Architectural Era
(It's Time for a Complete Rewrite)

Michael Stonebraker
Samuel Madden
Daniel J. Abadi
Stavros Harizopoulos
MIT CSAIL

Nabil Hachem
AvantGarde Consulting, LLC

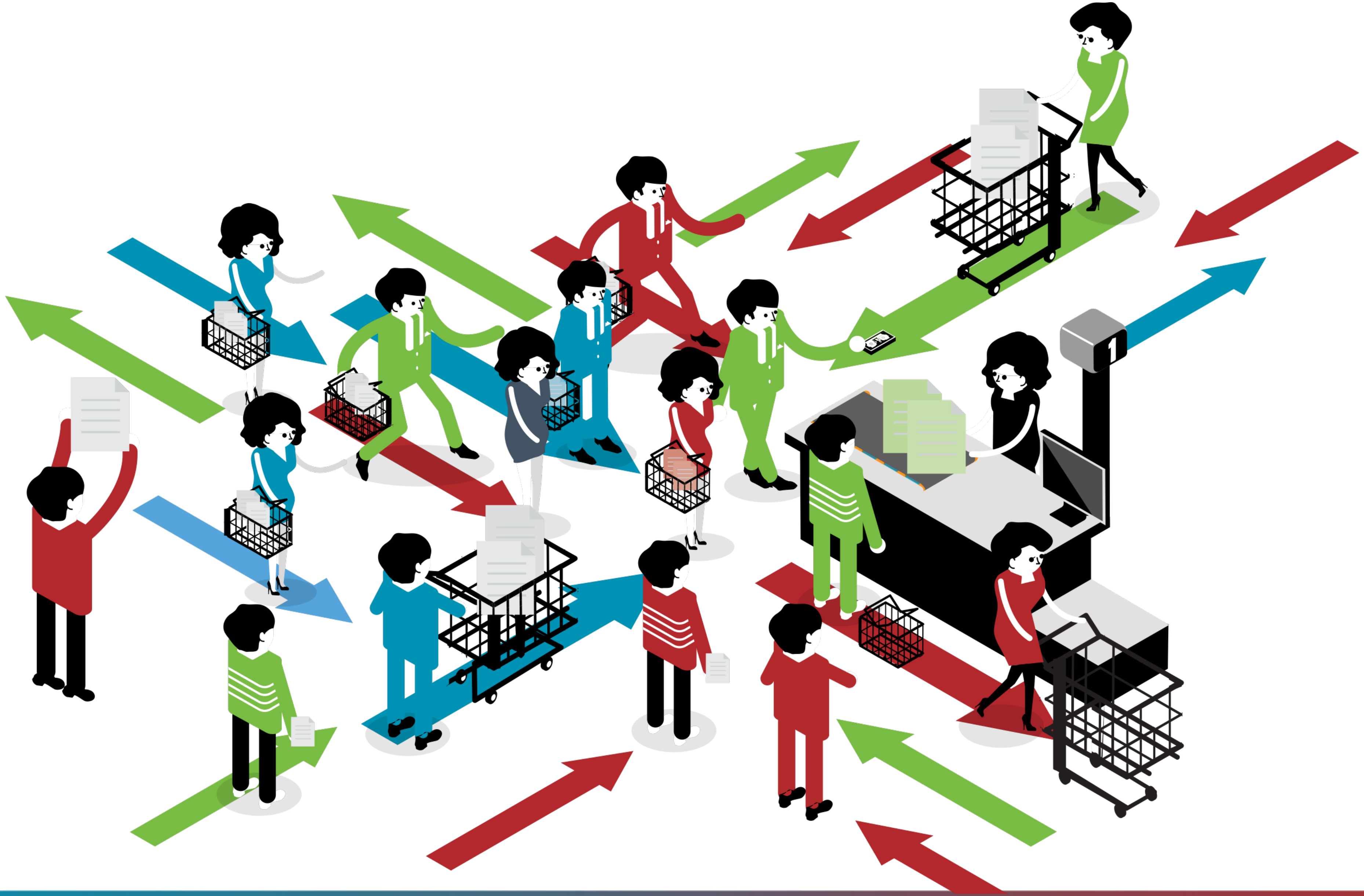
OLTP Through the Looking Glass, and What We Found There

Stavros Harizopoulos, HP Labs, Palo Alto, CA, stavros@hp.com
Daniel J. Abadi, Yale University, New Haven, CT, dna@cs.yale.edu
Samuel Madden, Michael Stonebraker, Massachusetts Institute of Technology, Cambridge, MA, {madden, stonebraker}@csail.mit.edu



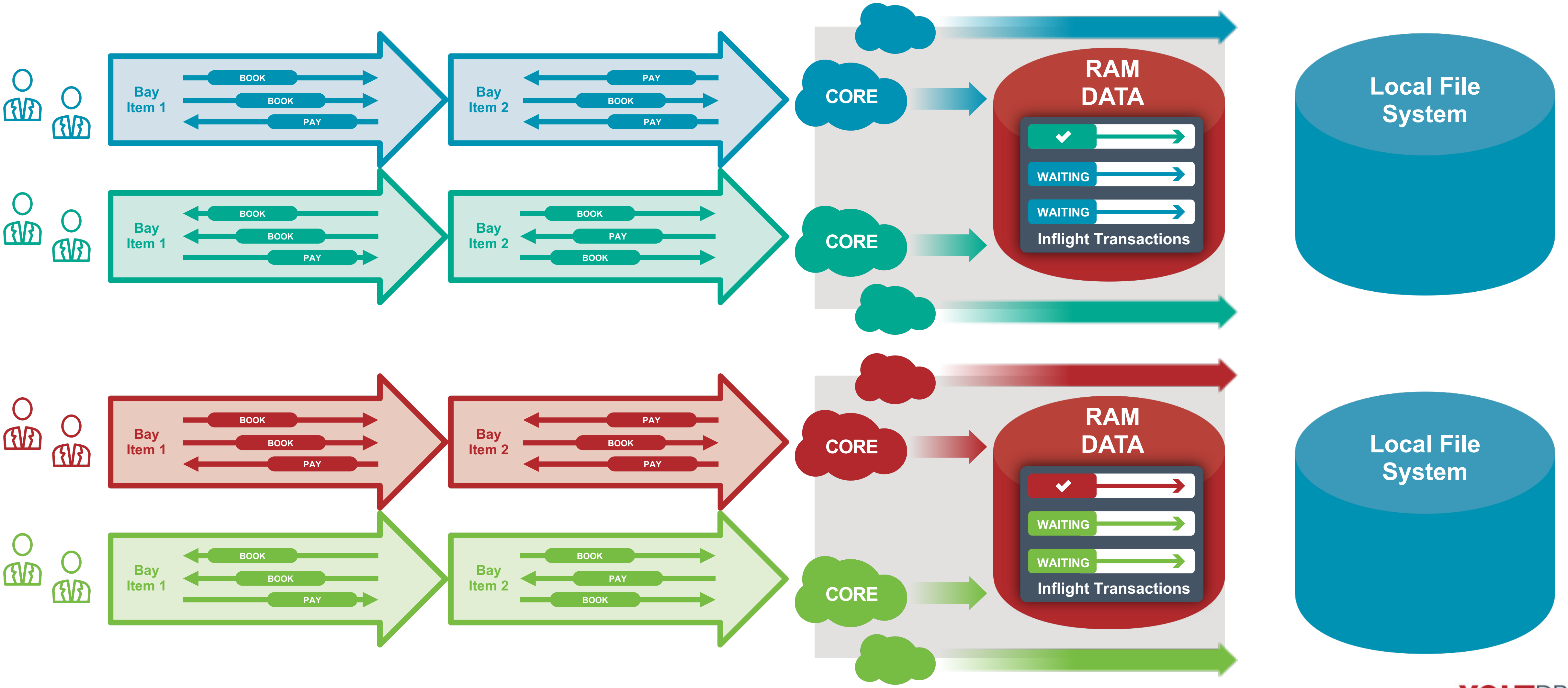
VOLTDB

If we tried this in a supermarket…



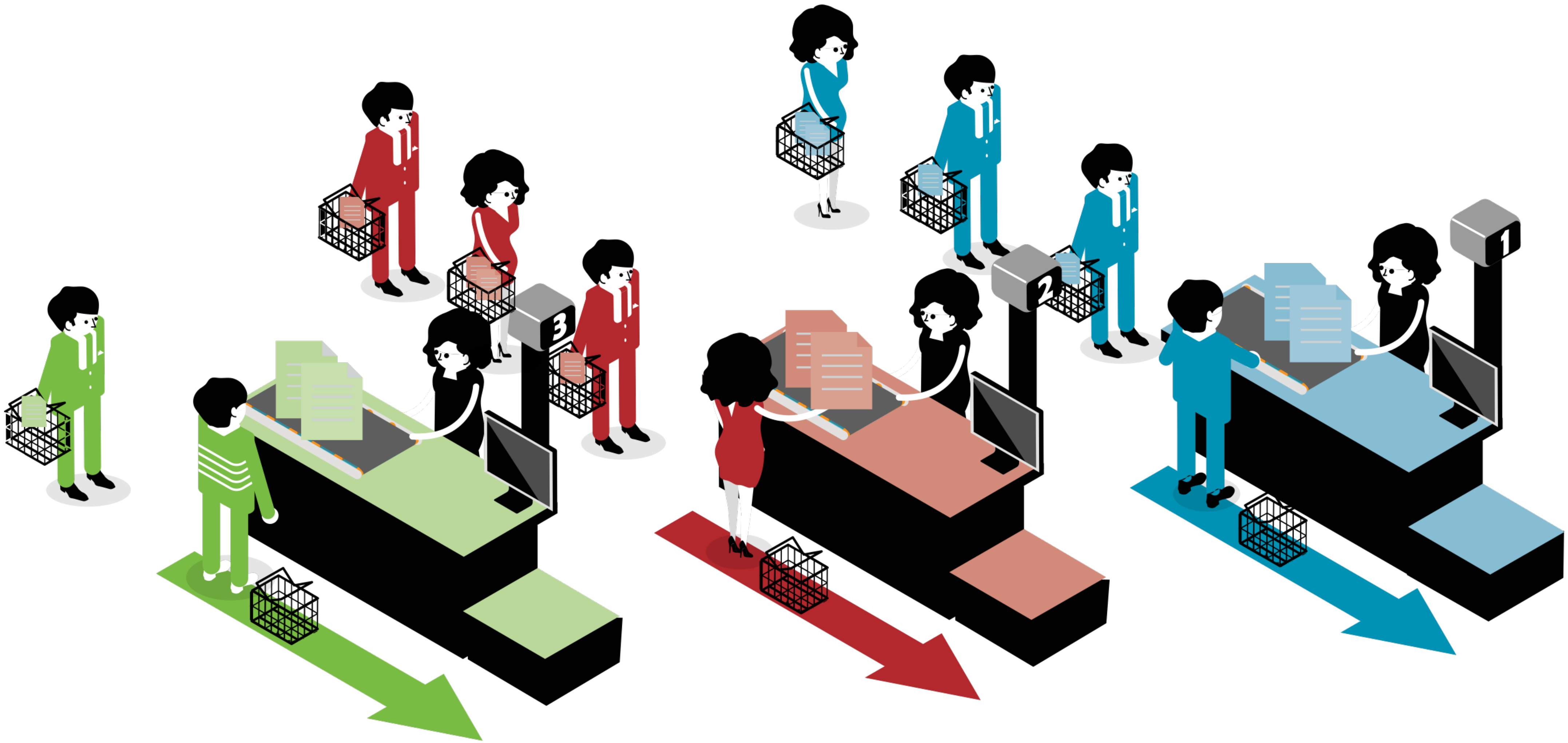
VOLTDB

How VoltDB works



VOLTDB

How a supermarket works…



VOLTDB

The only 3 ways to interact with any database

Approach	Examples	Strengths	Weaknesses
Many SQL Statements + Commit or Rollback	JDBC, ODBC,	Liked by developers, initial development is rapid	• Doesn't handle scaling OLTP loads well – DB spends its time figuring out who can see what instead of working • Constant locking problems for shared, finite resources • Failure of a client to Commit or Rollback causes a temporary resource leak
Move all the data to the client and back again	NoSQL, KV Stores	Very developer friendly	• Multiple updated copies of the data can arrive at the same time for scaling OLTP loads • All of the data gets moved across the network, every time.
Stored Procedures	VoltDB, PL/SQL	Predictable speed and best possible scaling characteristics	• Not in fashion with developers. • PL/SQL created perception of complexity. • Other implementations of Java Stored Procedures really slow.

VOLTDB

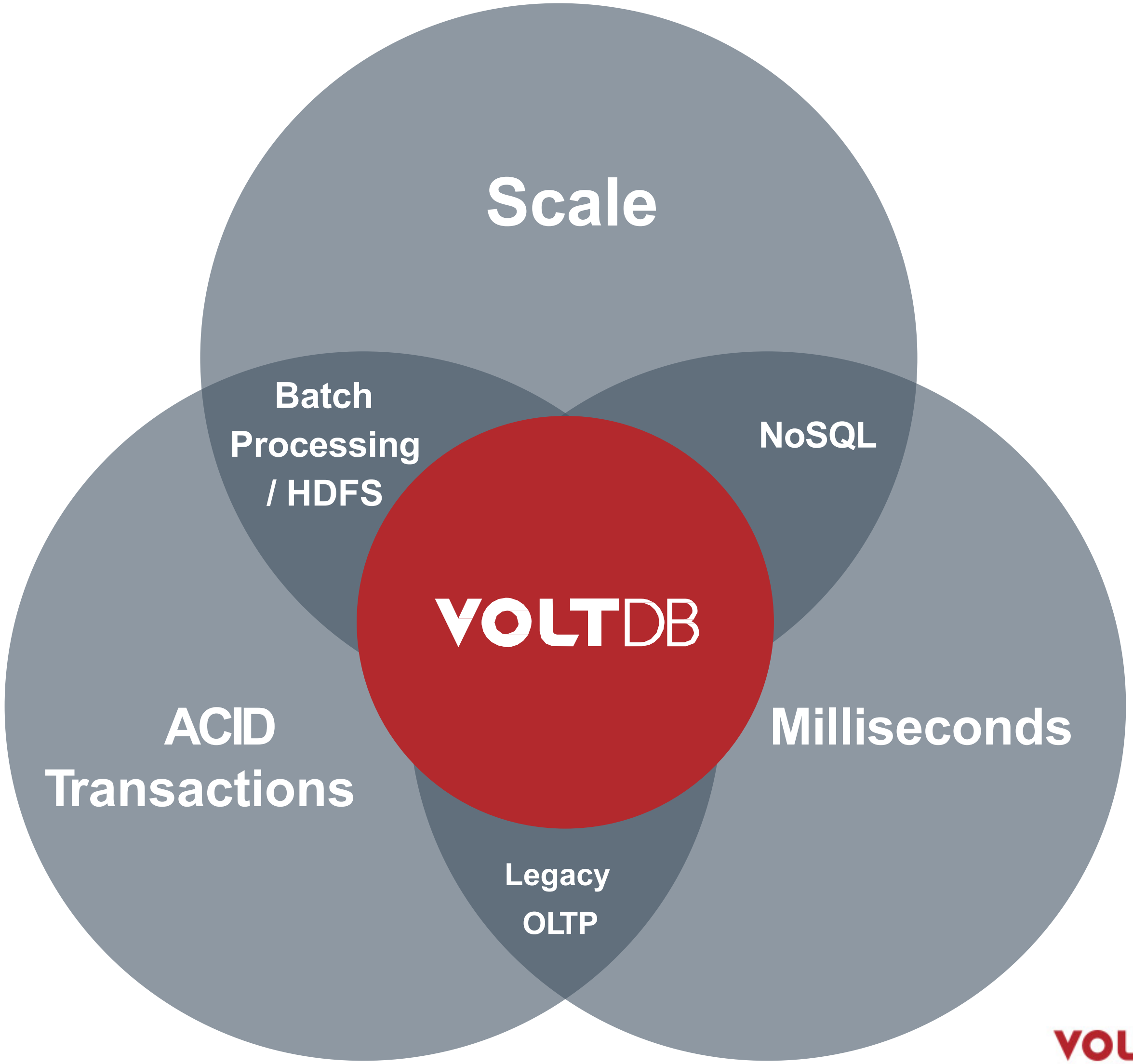
© 2014 VoltDB PROPRIETARY

VoltDB and Machine Learning

ML: The search for value

ML adds *real* value when:

- Combined with state
  - State can be a composite of multiple data sources
- Is used to inform and influence decisions
- Happens in real time
- Happens at scale



VOLTDB

VoltDB + ML

- The Good News
  - VoltDB appears to work with any ML engine with a Java runtime
  - VoltDB can be made to work with any ML engine with a C++ runtime
    - Requires JNI expertise
- Caveats
  - Runtime engine can’t have runtime dependencies/speak to another server
  - Runtime engine must be deterministic
  - Runtime engine must be fast (< 5ms)
    - >5ms undermines utility of using VoltDB
    - Be wary of instantiation costs

VoltDB + ML

- Limitations
  - Non Deterministic code
  - Other servers
  - Not C++ or JVM compatible
    - Not a showstopper, but…
  - Slow

- Example: Neural Nets
  - But GPDR may make them unusable in EU anyway…

ML Example – User Defined Function in H20

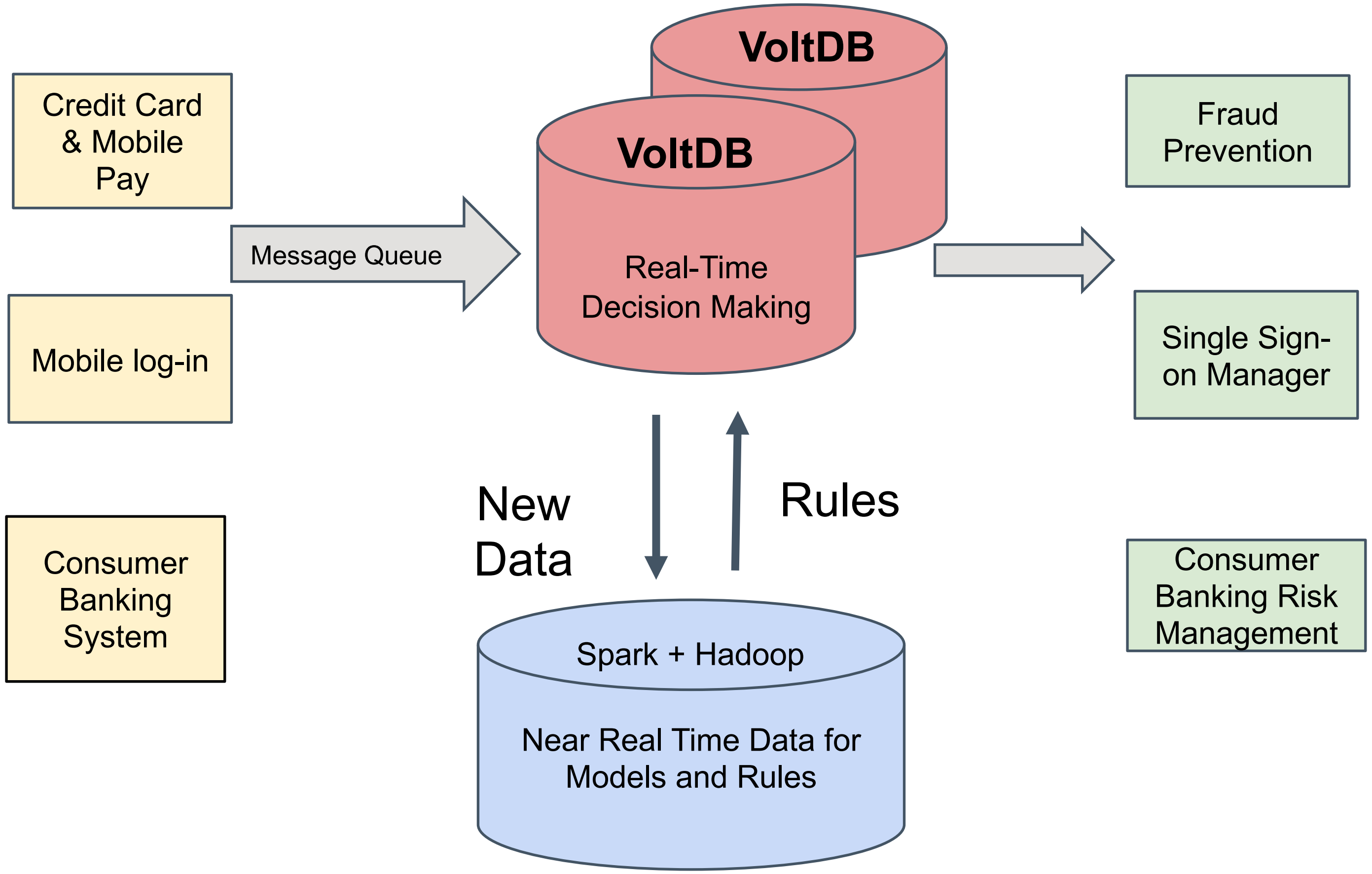
```java
public class AirlineDemoUDF {

    private static String modelClassName = "gbm_pojo_test";

    public String ademo(String cRSDepTime, String year, String month, String dayOfMonth, String dayOfWeek,
            String uniqueCarrier, String origin, String dest) {

        try {

            hex.genmodel.GenModel rawModel;
            rawModel = (hex.genmodel.GenModel) Class.forName(modelClassName).newInstance();
            EasyPredictModelWrapper model = new EasyPredictModelWrapper(rawModel);

            RowData row = new RowData();
            row.put("Year", year);
            row.put("Month", month);
            row.put("DayofMonth", dayOfMonth);
            row.put("DayOfWeek", dayOfWeek);
            row.put("CRSDepTime", cRSDepTime);
            row.put("UniqueCarrier", uniqueCarrier);
            row.put("Origin", origin);
            row.put("Dest", dest);
            BinomialModelPrediction p = model.predictBinomial(row);

            return (p.label);

        } catch (Exception e) {

            System.err.println(e.getMessage());
            return null;

        }

    }
```

```sql
CREATE FUNCTION ademo FROM METHOD h20.AirlineDemoUDF.ademo;

CREATE PROCEDURE flight_hist
PARTITION ON TABLE   flights COLUMN  f_FlightNum AS
SELECT f_cRSDepTime,  f_year,  f_month,  f_dayOfMonth,
f_dayOfWeek, f_uniqueCarrier,  f_origin,  f_dest
,ademo(f_cRSDepTime,  f_year,  f_month,  f_dayOfMonth,
f_dayOfWeek, f_uniqueCarrier,  f_origin,  f_dest ) ademo
from flights
where f_FlightNum = ?
order by f_year,  f_month,  f_dayOfMonth,f_cRSDepTime;
```

ML Example – Calling JPMML from a Procedure

```java
public VoltTable[] runModel(String pmmlFileName, VoltTable inputParams) throws Exception {

    Evaluator evaluator = pmmlEvaluators.get(pmmlFileName);

    if (evaluator == null) {
        throw new Exception("Model " + pmmlFileName + " not found");
    }

    List<InputField> inputFields = evaluator.getInputFields();
    Map<FieldName, FieldValue> arguments = new LinkedHashMap<FieldName, FieldValue>();

    // Sanity check input params

    if (inputParams == null) {
        throw new Exception("VoltTable inputParams can't be null");
    }

    if (inputParams.getRowCount() != 1) {
        throw new Exception("VoltTable inputParams must have one row");
    }

    if (inputParams.getColumnCount() != inputFields.size()) {
        throw new Exception("VoltTable inputParams must match length of inputFields. inputParams
                + inputParams.getColumnCount() + " columns, expect " + inputFields.size());
    }

    inputParams.advanceRow();
    for (InputField inputField : inputFields) {
        mapVoltparamToPmmlParam(inputParams, arguments, inputField);
    }

    Map<FieldName, ?> result = evaluator.evaluate(arguments);

    // Processing results
    // Retrieving the values of target fields (ie. primary results):
    List<TargetField> targetFields = evaluator.getTargetFields();
    VoltTable resultTable = mapPmmlTargetFieldsToVoltTable(result, targetFields);

    // other fields
    List<OutputField> outputFields = evaluator.getOutputFields();
    VoltTable otherTable = mapPmmlOutputFieldsToVoltTable(result, outputFields);

    VoltTable[] outputParams = { resultTable, otherTable };

    return outputParams;

}
```

```java
public class GolfDemo extends VoltProcedure {

    public VoltTable[] run(double temperature, double humidity,
            String windy, String outlook) throws VoltAbortException {

        VoltTable[] pmmlOut;

        try {

            JPMMLImpl i = JPMMLImpl.getInstance();
            VoltDBJPMMLWrangler w = i.getPool().borrowObject();
            final String modelName = "tree.model";
            VoltTable paramtable = w.getEmptyTable(modelName);
            paramtable.addRow(temperature, humidity, windy, outlook);
            pmmlOut = w.runModel(modelName, paramtable);

        } catch (Exception e) {

            System.err.println(e.getMessage());
            throw new VoltAbortException(e);

        }

        voltExecuteSQL(true);
        return pmmlOut;
    }

}
```



Application/Use Case

- Fraud Prevention
- Single sign-in of all Huawei phones
- Consumer banking risk management

Why VoltDB?

- > 50% reduction in fraud cases
- > $15M/year saved from fraud loss
- 10k complex Transactions Per Second
- 99.99% transactions finish < 50ms
- 10x better performance than traditional fraud detection

A Proven and Reliable Partner

Telco

Billing/rights management, subscriber data, etc.

Financial Services

Risk, market data management, customer mgt.

Personalize, Customize, Target

Ad optimization, audience segmenting, customer service

IoT Platforms, Energy, Sensor

Smart grid/meters, asset tracing & management

Infrastructure, Dashboards, KPIs

Data pipeline, system performance, streaming ETL.

VOLTDB