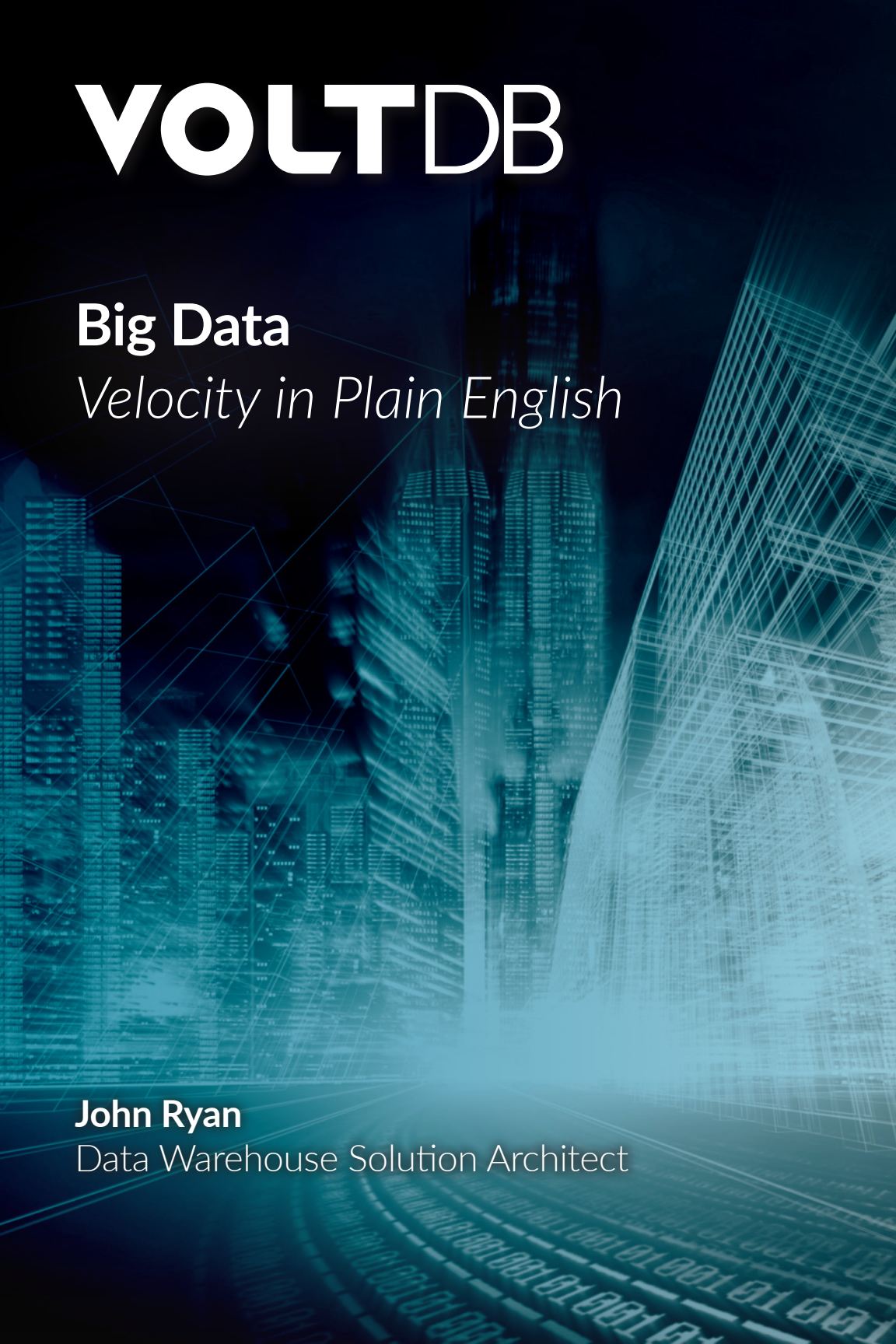
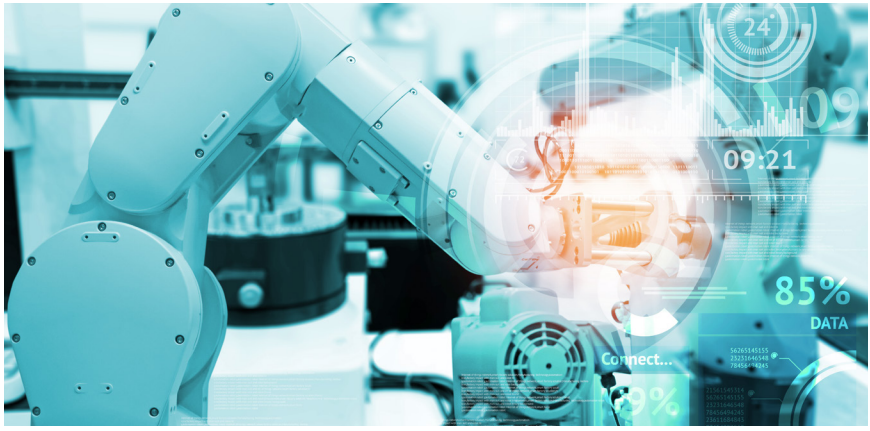

VOLTDB
Big Data
Velocity in Plain English
John Ryan
Data Warehouse Solution Architect

# Table of Contents

## The Requirement

The assumed requirement is the ability to capture, transform and analyse data at potentially massive velocity in real time. This involves capturing data from millions of customers or electronic sensors, and transforming and storing the results for real time analysis on dashboards. The solution must minimise latency — the delay between a real world event and it's impact upon a dashboard, to under a second.




Typical applications include:

- **Monitoring Machine Sensors**: Using embedded sensors in industrial machines or vehicles — typically referred to as The Internet of Things (IoT) . For example Progressive Insurance use real time speed and vehicle braking data to help classify accident risk and deliver appropriate discounts. Similar technology is used by logistics giant FedEx which uses SenseAware technology to provide near real-time parcel tracking.

- **Fraud Detection**: To assess the risk of credit card fraud prior to authorising or declining the transaction. This can be based upon a simple report of a lost or stolen card, or more likely, an analysis of aggregate spending behaviour, aligned with machine learning techniques.

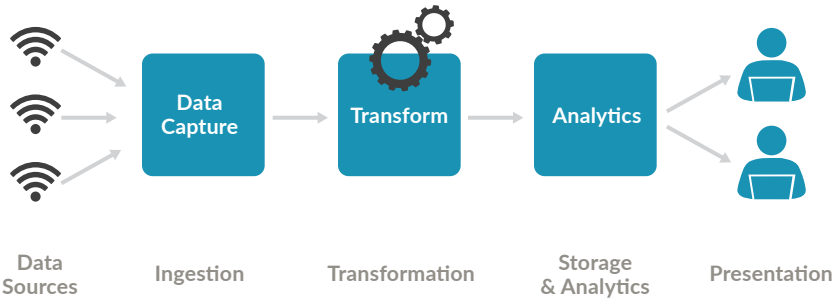
- **Clickstream Analysis**: Producing real time analysis of user web site clicks to dynamically deliver pages, recommended products or services, or deliver individually targeted advertising.

## What's the Problem?

The primary challenge for real time systems architects is the potentially massive throughput required which could exceed a million transactions per second. The relational database solutions from Oracle, IBM and Microsoft simply can't reach this level of throughput. Likewise, NoSQL databases can handle the data velocity, but have the disadvantages associated with a lack of SQL access, no transaction support and eventual consistency. Finally they don't support flexible join operations, and analytic query options are limited or non-existent. This means you can quickly retrieve a key-value pair for an event, but analysing the data is a challenge.

However, it doesn't stop there.

### Data Processing Pipeline



# Components Needed

The diagram above illustrates the main architectural components needed to solve this problem. This includes:

## Data Capture

- **High Velocity Data Transfer**: The ability to capture data, and handle high velocity message streams from multiple data sources in the range of hundreds of megabytes per second.
- **Message Queuing**: We can expect short term spikes in data volume which implies a message handling solution to ease the spikes, avoiding the need to scale up the entire solution for the worst possible case.
- **Guaranteed message delivery**: Which implies a fault tolerant, highly available solution that gracefully handles individual node failure, and guarantees message delivery
- **Architectural Separation**: To decouple the source systems from the messaging, transformation and data storage components. Ideally the solution should allow independent scaling of each component in the stack.
- **A Range of adapters and interfaces**: To support multiple feeder systems and sensors, configurable at run time while avoiding the need for system down-time.

## Transformation

- **In memory streaming**: The need to reduce latency implies an in-memory data streaming and transformation solution with data restructured and transformed in real time.
- **Data integration**: The transformation process will almost certainly need to combine transaction streams with existing reference data from existing databases and other (eg. Hadoop and NoSQL) data sources. The solution must therefore provide excellent data source connectivity.

## Storage and Analytics

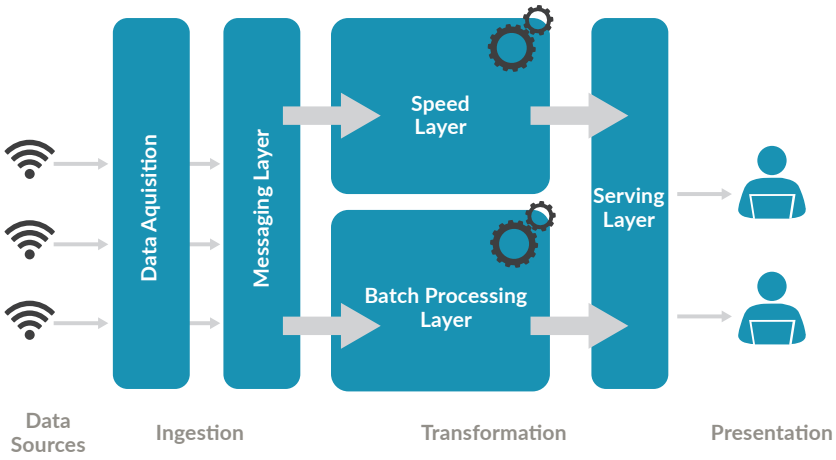
- **High Velocity Ingestion**: The data storage solution must be capable of accepting millions of transactions per second, ideally accessible via industry standard SQL, and with full ACID transaction support.
- **Data Analytics Solution**: Again with full SQL support and the ability to support both geo-location and real time analytic style queries without blocking data ingestion.
- **Dashboard Connectivity**: The solution must provide support for open connectivity standards including JDBC and ODBC to support Business Intelligence and dashboards.

Thankfully, there are battle hardened tools available (many open source) which are already proven in real-world cases against massive data volumes.

These include:

- **Apache Flume**: For web data extraction and ingestion (optional)
- **Apache Kafka**: A massively scalable data streaming and middleware solution with guaranteed message delivery
- **Apache Spark Streaming**: For near real time data transformation and streaming
- **VoltDB or MemSQL**: For high velocity data capture and real-time analytics

The Lambda Architecture




## The Traditional Solution

The diagram above illustrates a common architecture referred to as the Lambda Architecture which includes a Speed Layer to process data in real time with a Batch Layer to produce an accurate historical record. In essence, this splits the problem into two distinct components, and the results are combined at query time in the Serving Layer to deliver results to the user.

> *Keeping code written in two different systems perfectly in sync was really, really hard.*
>
> *— Jay Kreps on Lambda (LinkedIn)*

While the Lambda Architecture has many advantages including decoupling and separation of responsibility, it also has the following disadvantages:

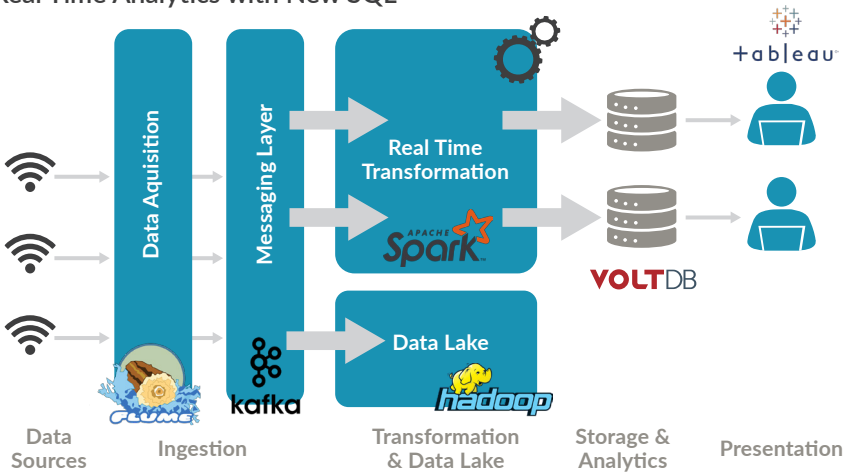
- **Logic Duplication**: Much of the logic to transform the data is duplicated in both the Speed and Batch layers. This adds to the system complexity, and creates challenges for maintenance as code needs to be maintained in two places – often using two different technologies.
- **Batch Processing Effort**: The batch processing layer assumes all input data is re-processed every time. This has the advantage of guaranteeing accuracy as code changes are applied to the data every time, but potentially places a huge unnecessary batch processing burden on the system.

- **Serving Layer Complexity**: As data is independently processed by the Batch and Speed layers, the Serving Layer must execute queries against two data sources, and combine real time and historical results into a single query. This adds additional complexity to the solution, and may rule out direct access from some dashboard tools or need additional development effort to support end user queries.
- **NoSQL Data Storage**: While batch processing typically uses Hadoop/HDFS for data storage, the Speed Layer needs fast random access to data, and typically uses a NoSQL database, for example HBase. This comes with huge disadvantages including no industry standard SQL interface, a lack of join operations, and no support for ad-hoc analytic queries.

When the only transformation tool available was Map Reduce with NoSQL for data storage, the Lambda Architecture was a sensible solution, and it has been successfully deployed at scale at Twitter and LinkedIn. However, there are more advanced (and simple) alternatives available.

# The NewSQL Based Solution

Real Time Analytics with New SQL




The diagram above illustrates an alternative solution with a single real time data flow from source to dashboard. The critical component that makes this possible is the NewSQL database technology (eg. VoltDB, NuoDB or MemSQL) which supports full ACID consistency while processing millions of transactions per second.

The components in the above solution are:

- **Apache Flume**: An optional component for high throughput data capture of web logs for clickstream analysis.
- **Apache Kafka**: For fault tolerant message queuing and data broadcast.
- **Apache Spark Streaming**: For near real time in memory data processing and transformation. Also consider Apache Storm or Flink.
- **Hadoop / HDFS**: An optional component for long term inexpensive storage, and a Data Lake.
- **VoltDB**: For real time data ingestion and storage at millisecond latency in addition to real time analytics. Also consider MemSQL, NuoDB and CockroachDB.
- **Tableau**: For analytic presentation and dashboards.

The advantages of this architecture are:

- **Transformation Simplicity**: With all data transformation logic in the Spark Streaming component (using industry standard SQL), there's no code duplication or multiple technologies to cause maintenance issues.
- **Real Time Accuracy**: As the database solution provides full relational support and ACID compliance at millions of transactions per second, there's no issues associated with eventual consistency from NoSQL solutions.
- **Analytic Simplicity**: In common with many NewSQL databases, VoltDB supports real time analytics using industry standard SQL which is simply not possible on NoSQL solutions. In addition, dashboard users (for example Tableau), can directly connect to the database, and seamlessly query results without the need to combine data from multiple sources. This compares well to NoSQL solutions where the database design is far less flexible than the NewSQL relational design.

Of course any real time solution must fit into an existing batch oriented architecture including integration into a data lake, and the solution includes an additional feed into data into Hadoop HDFS for subsequent long term storage and batch processing.

## The NewSQL Advantage

The technology component that really makes this architecture possible, is the addition of a hybrid real time and analytics database, the NewSQL database. First proposed by Dr Michael Stonebraker in his paper ***The End of an Architectural Era*** this provides a database platform redesigned from scratch to process millions of transactions per second on a horizontally scalable hardware platform.

> *Modern transactional databases overwhelmingly don't operate under textbook "ACID" isolation*
>
> *— Dr Peter Bailis. University of Stanford.*

Running almost entirely in memory, NewSQL databases stand out for their ability to meet or exceed the processing capability of NoSQL databases, but with the significant advantages of:-

- **A Fully relational database**: Complete with join operations, analytic functions and full support for industry standard SQL. This provides a much more flexible query solution than the NoSQL alternatives.
- **Full ACID compliance**: All NewSQL databases fully support transactions, and one (VoltDB) even exceeds the isolation level provided by Oracle to provide full serializability. This compares well to NoSQL databases that provide a very basic level of Eventual Consistency.
- **Millisecond Latency**: As data is processed in memory, these databases often average around two milliseconds for write operations, and scale to millions of transactions per second. This compares well to the standard database alternatives from Oracle or Microsoft that peak at thousands of transactions per second.
- **Fault Tolerance**: As data is replicated to two or more in memory servers in a horizontally scalable architecture, these solutions are purpose built for 24x7 operation. Some solutions (eg. MemSQL and NuoDB) can independently scale the processing and storage servers for additional flexibility.

- **On Premises and Cloud**: Most NewSQL databases can be deployed on premises, on dedicated or virtual machines, or in the cloud on Amazon, Google or Microsoft services. This can be huge advantage for IT departments not quite ready to go entirely cloud based.

## Thank You

Thank you for reading this far. If you found this interesting, do follow me for similar articles in future.

If you're curious about how NewSQL databases compare to traditional and NoSQL solutions, you may want to read my other article, Database Technology 3: NewSQL in Plain English.

## About the Author John Ryan

After 30 years as a freelance Data Warehouse Architect, I'm turning my attention to Big Data including Hadoop, NoSQL and NewSQL.

Follow me on:


twitter.com/@DatabaseCafe


linkedin.com/in/johnryanuk/