VOLTDB
SMART DATA FAST.™

WHITE PAPER

THE DEVELOPER GUIDE TO STREAMING DATA APPLICATIONS

Successfully writing Fast Data applications to manage data generated from mobile, smart devices and social interactions, and the Internet is development's next big challenge. The availability and abundance of fast, streaming data presents an enormous opportunity to write smart applications that extract intelligence, provide insight, and make everyday products, services, places, farms, cities, grids, buildings, and homes a source of intelligence.

Modern applications need to manage and drive value from fast-moving streams of data. But traditional tools such as conventional database systems are simply too slow to ingest data, analyze it in real-time, and make decisions. They can't meet Fast Data's demands. Successfully interacting with Fast Data requires a new approach to handling these new data streams.

As Fast Data emerges as a required component of a complete data-at-scale stack, several technologies are being proposed as possible solution components. These fall into three categories: fast OLAP systems (the province of Business Intelligence applications), stream-processing systems, and OLTP (database) systems. Each of these solutions can be highly capable but some are better suited to Fast Data than others. Organizing the Fast Data contenders by their core architecture types provides a way to evaluate their core strengths and weaknesses for the requirements of Fast Data applications.

Fast OLAP

OLAP solutions enable fast queries against raw (structured) data. Eliminating the need to "organize/accumulate" (by time window, session, or volume) is a primary part of their value proposition. They organize data at query time, not at ingest; mature OLAP systems are very good at doing so.

OLAP vendors believe the most valuable part of Fast Data is reducing time-to-reporting and enabling real-time BI. Vendors talk about the benefits of using columnar compression to store large amounts of data (years of history vs. hours) in memory; they also emphasize query speed.

Fast OLAP systems organize data to enable efficient queries across multiple dimensions of terabytes to petabytes of stored data. Typically these systems organize data in a fashion that allows cheap, fast, effective compression (e.g., run-length-encoding). Compression is usually critical: it allows the OLAP system to store more data in the same storage footprint, and it lets the OLAP system scan many entries while minimizing expensive disk accesses.

Where OLAP solutions fall short in Fast Data use cases is in transactional (multi-factor) decisions. OLAP offerings are analytics engines, not transaction processing engines. As noted, vendors highlight compression – the ability to store large amounts of data. They de-emphasize integration with competing OLAP systems (export/archive).

"Where OLAP (multi-dimensional) solutions fall short in Fast Data use cases are in transactional (multi-factor) decisions."

Stream processing

Streaming systems' main purpose is to capture data. Unlike OLAP and OLTP systems, streaming systems are not optimized to store data, nor do they optimize for fast record lookup. And since they aren't storing data, they are not optimized for scans across different dimensions of the data set. Instead, streaming systems are optimized for running computations across a "stream" of arriving events. Almost all offerings in this category organize a set of functions and run those functions against moving windows. Functions can be arranged in parallel (run input x against f(x) and g(x)), or in serial (run input x against f(x) and then run g() against the output: g(f(x))). These systems are very good at scaling **pre-defined** real-time analytics against Fast Data sources. The ability to compose these computations also enables different real-time ETL applications.

Some streaming proponents believe the most valuable part of Fast Data is scalable message processing and coordination between systems. Vendors and popular open source stream processing projects promote strengths in data integration and message pipelines.

Complex Event Processing (CEP) and streaming solutions are less-than-ideal choices for operations that require state. They are primarily on-ramps to OLAP. While streaming/CEP solutions can accomplish streaming analytics, they are not sufficient, without a supporting back-end database, for applications that rely on decisions or ETL. As a result, CEP/streaming offerings are often configured with a back-end database to address the 'fast decisions' use case. However, bolting on a back-end database will result in lower performance and higher latency than a fast OLTP solution.

"Streaming systems are primarily on-ramps to OLAP."

Fast OLTP

Fast OLTP vendors believe that per-event decision-making (requiring ACID semantics and database transactions), real-time data enrichment, and streaming analytics are critical when building smart, fast applications.

OLTP systems organize data as a series of records, where records may be "rows" or "documents". They are durable systems and can persist user data across failures. OLTP systems are designed for fast record lookups using indexes, and provide a query/transaction model that allows applications to query and read/write record data coherently and consistently. OLTP systems are typically designed to make writes to a specific record (or field) efficient and safe, but are not designed to scale multi-dimensional reads of large amounts of records like an OLAP offering.

Within OLTP systems, there are two types of architectures: traditional SQL systems and the New/NoSQL systems.

Traditional SQL systems are disk-based systems that can be challenging to scale at the throughput required by today's Fast Data requirements. These systems are more general-purpose systems.

NewSQL and NoSQL solutions can provide the speed and availability required by Fast Data applications. Each comes with its own specialty. NoSQL systems trade off query expressiveness (SQL) and schema for a flexible data model, low-latency lookup, and high availability. NewSQL solutions provide similar scalability but specialize this architecture, providing the expressiveness of SQL queries, strong consistency, and high availability, while providing a strong schema contract.

Use case vs. contender: Mapping the landscape

FAST OLAP	CEP/STREAMING	FAST OLTP
• Cannot generate real-time responses and decisions. • An evolution of OLAP capability to in-memory; not a sustainable strategic value-add vs. columnar incumbents (Vertica, Redshift, etc.) • May have poor SQL support – not a substitute to incumbent column stores (MemSQL)	• Good at data capture. • Good at ingest to OLAP and pre-defined read-only analytics. • Other operations introduce complexity and lack of reliability. May require development work; needs a back-end database.	• Fast Data is message/event oriented and databases are query/transaction oriented. Put messages first and support with DB where necessary. • Suitable for analytics with pre-defined queries. • No compression: too expensive to store large datasets. May require complex OLAP integrations.

Three drawbacks of streaming solutions

”In the rush to develop applications for Fast Data, developers focus on the stream of data, not on the desired business result.”

Streaming solutions lack context

Streaming solutions can ingest fast-moving data feeds but they lack context and state, both necessary to support decision-making. For example, filter, aggregate, and join operations (aka enrich) require state. Streaming systems thus must be complemented by back-end databases; as standalone offerings their value is primarily focused on fast ingestion. Their poor query support inhibits interactivity and introduces network I/O round trips to the back end, where they still need fast back-end DBs. Steaming solutions thus are a compromised solution for Fast Data applications that rely on context.

Streaming solutions are not architected for real-time decisions

Decisions are fundamental ACID workloads. Except for the scale (required performance) load created by M2M/IoT/Web applications, these are OLTP applications. They require all of the requirements of an ACID system: atomicity of side effects, consistency in evaluation and constraint checks, isolation from other concurrent transactions, and durability of results. Streaming systems are not designed to offer fast per-event responses to applications. The strength of streaming systems is algorithmic processing of windowed data. Streaming systems do not implement ACID transactions and are not designed to be durable records. Lacking standard application interfaces (ODBC/JDBC) or broad ad-hoc query capability, streaming solutions are less than ideal for real-time decision use cases. Apache Spark, an extension of the OLAP Hadoop warehouse, is an analytics tool that reduces time to analytics (vs. batch processing/periodic reporting). However, Spark does not support decisions.

Streaming solutions lack operational transparency

Streaming solutions can only be queried for their statically-configured topological results; streaming solutions that maintain fixed aggregations need to store those aggregations in fault tolerant (durable) back-ends, introducing the complexity of another storage system.

Fast data adoption in agile development: Inflection points

With Fast Data, things that were not possible before become achievable.

As more developers are tasked with building applications to handle fast streams of data, providers of streaming solutions are reaching an inflection point. Unable to meet the full-stack demands of application developers, they are turning to 'composed' solutions, e.g., Kafka+Storm+Spark+NoSQL database; the Lambda architecture, which splits the functions of batch, speed and serving; or highly customized, purpose-built solutions, cobbled together from open source projects and custom code. These approaches rely on the 'cool' technology du jour, sacrificing enterprise scale, proven, high-quality code, and repeatability. In the rush to develop applications for Fast Data, developers focus on the stream of data, not on the desired business result.

Solving the Shortcomings of Streaming Solutions

Understanding the promise and value of Fast Data requires an understanding of the nature, utility, and shortcomings of streaming solutions.

While much can be accomplished by ingesting fast-moving streams of data, two of the three contenders for streaming solutions lack key functionality necessary to build applications that enable businesses to act in real time as data flows into the organization from millions of endpoints: sensors, mobile devices, connected systems and the Internet of Things.

Yes, fast and big data have different requirements, and it's necessary to have a component on the front end of the data pipeline to manage streams. Yet how much more effective would it be to have the ability to handle streams of data by ingesting and interacting on the data stream; performing real-time analytics on the data in motion; and making data-driven decisions on each event in the stream? In this model, applications can take action on streams of data, and processed data can be exported at high speed to the data warehouse for historical analytics, reporting, analysis, and more.

Streaming solutions, while appealing for fast ingest, do not provide the missing link between fast streams of data and Fast Data applications. Application developers must be free to write code that adds value to the organization, rather than being burdened by writing code to manage streams of data as it flows through the pipeline. An in-memory operational system that can decide, analyze and serve results at Fast Data's speed is the answer to handling fast streams of data at enterprise scale.

Fast Data applications give organizations the tools to process high volume streams of data while enabling millions of complex decisions in real-time. With Fast Data, things that were not possible before become achievable: instant decisions can be made on real-time data to drive sales, connect with customers, inform business processes, and create value.

Next Steps

To learn more about VoltDB, visit www.VoltDB.com.
Product documentation, developer support forums and an open source version of VoltDB are freely available.