TECHNICAL PAPER

Stateful Stream Processing vs. Legacy Stream Processing Dissecting the Differences

Introduction

Hyper-connected applications, things/sensors, and humans are producing more data today than ever before. Data has become the new oil for the Digital Economy. Big Data can drive significant business benefits if mined, analyzed and acted upon effectively. Case in point are the FANG Companies (i.e. Facebook, Amazon, Netflix, and Google), that have become the most valuable companies in the world worth billions of dollars and dominate the tech industry by effectively managing Big Data.

Why Stream Processing is important now more than ever before?

With the emergence of 5G, Big Data is about to experience a seismic shift. 5G promises data rates 100x of 4G, network latency of under one millisecond, the ability to support one million devices/sq. km., and 99.999% availability of the network. The velocity and volume of new data under 5G is expected to grow exponentially. In addition to the rise in V's, the complexity and demands from operational analytics will become far greater. 5G is set to disrupt the way we ingest, store, and analyze data yet again.

5G is set to make the Internet of Things (IoT) a reality by providing fast connectivity and higher capacities. Gartner estimates: "IoT endpoints to reach an install base of 25.1 billion units by 2021." — gartner.com

Real-time actionable insights is the future in the era of IoT. Action oriented decision making while the data is fresh/in-motion is the next key differentiator for data driven organizations.

VOLTDB

These endpoints will result in a massive tsunami of data that will require real-time stream processing and action-oriented intelligent decisioning at very low latency. The low latency requirements of IoT devices and apps can only be met with micro data centers at the edge.

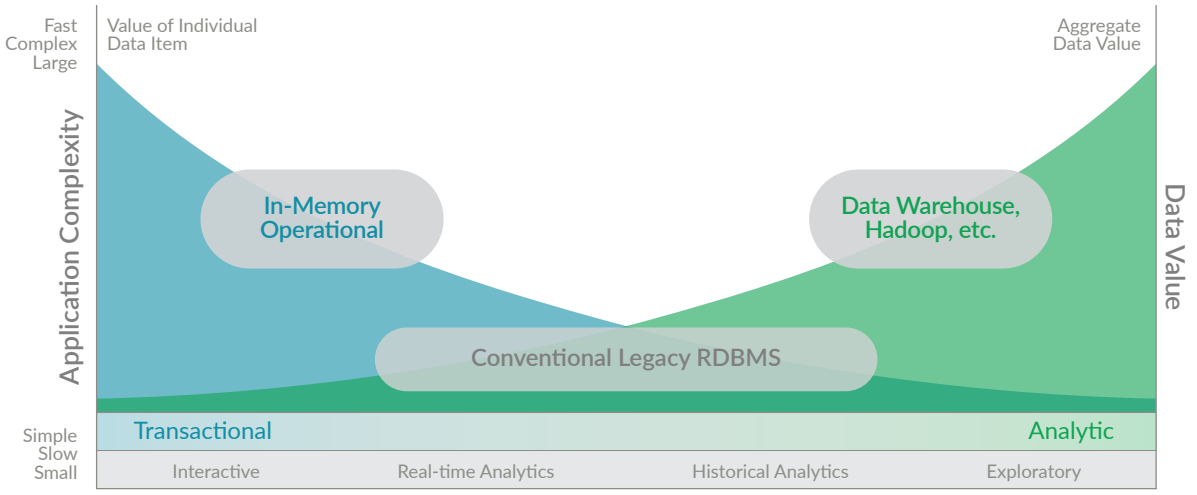


Figure 1: The Database Universe

As depicted in Figure 1, the business value of data diminishes as the data gets stale. This phenomenon is extenuated even further for IoT apps, where a lag of even a few milliseconds can lead to not just loss of revenue (in use cases such as: customer experience management, customer churn, fraud detection and others) but potentially even human life (use cases such as self-driving cars). Real-time actionable insights is the future in the era of IoT. Action-oriented decision making while the data is fresh/in-motion is the next key differentiator for data-driven organizations. Real-time now means milliseconds; businesses demand moving from post event reconciliation to in-line data processing. Value of data diminishes rapidly; if action is not taken immediately, the opportunity to monetize an event is lost.

The 5G network utilizes high frequency bands to deliver on the promised speed. However, high frequency bands cause considerably higher interception, which requires 100X more small cell towers than the number of macro towers. This increase in cell tower density leads to an exponential increase in the number of events from two specific functions: the Access Management Function (AMF) and the Session Management Function (SMF).

With a service-based architecture, data stores have been unified into two categories: structured subscriber data in a User Data Repository (UDR) and all other data in an unstructured storage function in Unstructured Data Storage Function (UDSF). These are extensions of the shared data storage concept that has become a mainstay with the genesis of Virtualized Network Functions (VNFs). Both of these functions require a scalable, in-memory stateful stream processing solution that not only performs, but also does not lose data and provides accurate answers and decisions. This takes the need beyond data storage and streaming data, and combines the needs into a unified platform.

The decisions being made in the modern system are no longer based off of static rules and policies – rules are continually evolving by learning from new data and training models. These learnings must be deployed into the decision-making process, and this needs to happen instantly, seamlessly, in a live environment, without any stoppage of service.

Limitations of Legacy Stream Processing

Wikipedia defines Event Stream Processing (ESP) as "Event stream processing, or ESP, is a set of technologies designed to assist the construction of event-driven information systems. ESP technologies include event visualization, event databases, event-driven middleware, and event processing languages, or complex event processing (CEP)."

While a message bus (also known as message queues) is defined by Wikipedia as "software-engineering components used for inter-process communication, or for inter-thread communication within the same process." Message bus technology enables subscribers to pull specific messages (or events) from publishers. After processing the message, it is either removed from the queue or persisted in some systems.

Stream processing solutions on the market today were originally architected as message queues. Their inherent architecture limits them from offering transactional and analytical processing capabilities. Below is an overview of modern Enterprise app requirements that are unmet by legacy stream processing solutions.

Legacy steam processing architectures think of data streams as either event time windows or process time windows to "batch" the data in order to process it. But reality does not operate in discrete time windows. Batch processing in time windows, results in either a loss of important events or considerably delayed data processing.

- **Responsiveness:** Mission critical apps often have someone or something critical waiting for a response on the analysis to take a decision and act on it.
- **Contextual state:** Is critical to making meaningful business decisions. Existing stream processing tools, at best, offer either static state used primarily for enrichment, or state that is isolated to an individual stream. This limits the processing to very basic data models. Most modern apps require decisioning that relies on complex data models. These apps also rely on low latency decisions, disk based stream processing tools that use a 3rd party database to store state that just can not meet low latency requirements.
- **Complexity:** Along with state, modern stream processing apps are required to process anywhere from tens to thousands of rules in real-time and apply embedded complex logic via Machine Learning (ML) to multiple incoming streams. Additionally, as discussed earlier, current stream processing platforms can not support complex data models, i.e. events that map to multiple tables.
- **Delivery Guarantees**: Ideally an "exactly once" guarantee is desired, i.e. you will see the event exactly once. An acceptable alternative is an "at-least once" guarantee with idempotent operations.
- **ACID:** Stream Processing technologies on the market today do not offer native ACID Transactions required for dealing with exactly once semantics. Lack of ACID is especially pronounced when scaling a cluster.
- **High Availability:** Current stream processing systems only support asynchronous replication across clusters. With asynchronous replication, the chances of losing vital business data is very high in the event of node failure. Synchronous replication, on the other hand, writes data to the primary node and a copy/copies to one/many nodes simultaneously, ensuring data safety at all times.

Legacy steam processing architectures think of data streams as either event time windows or process time windows to "batch" the data in order to process it. But reality does not operate in discrete time windows. Batch processing in time windows results in either a loss of important events or considerably delayed data processing.

The evolution from Lambda architecture to Kappa architecture addresses updated code reprocessing of the data without needing a batch layer, but still operates within the constraints of time windows. The binary view of time windows is unnatural due to the chunking of data processing. Additionally, current stream processing solutions can only process individual streams in conjunction with some static data, or process the data as an enrichment pipeline (even when done as a continuous stream). This is not suitable for the evolved streaming requirements in which the stream is not only processed for downstream applications, but also needs to drive decisions either per event or in a complex event processing frame of reference.

"Turning database technology inside out" involves a lot of compromise, as evidenced in current stream processing solutions attempting to retrofit a database and interact with the store. This approach sacrifices many guarantees like atomicity and consistency that one would expect and need from a data platform. Modern Enterprise apps require stream processing needs to go beyond the Kappa architecture.

VoltDB's Stateful Stream Processing

Streaming data can be categorized into three primary functions: ingestion, processing and storage. While there are disparate solutions for each of these functions - in a world that now requires low latency complex decisions at scale - you need a system that brings all these functions together. Dedicating a team of developers to write and maintain glue code is not a viable long term strategy that can be implemented in production.

While traditional stream processing solutions employ a "pipeline" approach, it is more often than not a decision making process oriented to initiate an action. Secondly, all new streaming data is ultimately going to be new training data for learning systems. When the learning iteration is complete, you need to bring that new and improved insight into your decision making process. VoltDB offers all of these essential capabilities into a single platform i.e. combining ingest, store, process/decide, notify/alert and import machine learning outcomes into a SQL accessible form. This confluence of capabilities is defined as the Stateful Stream Processing Architecture. The wide range of use cases that Stateful Stream Processing enables, can be classified under four categories; in the following section we will examine each category and examples that fall under each category.

VoltDB was born as an in-memory translytical database, and as it has evolved into a full fledged stream processing platform, its database roots enable VoltDB to offer stateful stream processing. VoltDB offers all the essential attributes that are critical for stateful stream processing:

- Blazing Fast:
  - **In-Memory:** VoltDB's database roots enable it to store state locally in-memory. Reading and writing data from/on disk is a laborious process. Memory, however, is much faster than disk/flash drives, and now more affordable than ever.
  - **Per-machine Efficiency:** VoltDB was architected for high throughput and low latency on smaller clusters.

- **Distributed:** VoltDB can be optimized for individual app requirements with partitioning. Database tables and the stored procedures that access those tables are partitioned on one or more machines enabling distributed state storage and processing.
  - **Scaling:** VoltDB's architecture enables it to scale seamlessly to increase (or decrease) the throughput of your app. Simply by increasing the number of nodes in your VoltDB cluster by either saving a snapshot of the database that can then be reloaded to the resized cluster, or by adding nodes while the database is running without interrupting production.
- **Fault Tolerance:** VoltDB is committed to keeping your valuable data safe, it keeps running in the face of most failures. With VoltDB's telecom and finance pedigree it has been trusted to run mission critical enterprise apps with 99.999% availability. VoltDB achieves industry best fault tolerance with the following features:
  - **High Availability:** VoltDB was built to withstand hardware, software, or network failures. It is currently the only stream processing and data management platform to offer both inter-cluster and intra-cluster replication.
  - **Full Disk Persistence:** VoltDB employs snapshots and command logs to achieve full disk persistence:
    - **Snapshots:** Are a complete record of your VoltDB data and app, taken at a single logical point in time and stored on the filesystems of your clustered machines. VoltDB offers serializable consistency; every single state can been seen as the serial application of all previous transactions. A point-in-time snapshot represents the state between two transactions, globally, across the whole cluster.
    - **Command Log:** VoltDB fills in the gaps between snapshots by writing a log of operations to disk as it processes. In the unlikely event of a cluster-wide failure, the system can be recovered by loading the most recent snapshot, then replaying the log from the snapshot's logical time to the end of the log. In this way, operations between snapshots can be stored safely on disk.
  - **Replication:** VoltDB offers passive (unidirectional, from master to replica) and cross datacenter replication (database copies reside in multiple data centers, and are actively updated across all clusters). Replication ensures that your data is safe in the event of a power failure or other unforeseen disasters.
  - **Strongest ACID State Guarantees:** VoltDB's partitions run independently, while providing ACID semantics for the commands processed from its associated command queue.
    - Atomicity–While executing the commands, the program maintains in-memory undo logs so that aborted commands can roll back, ensuring atomicity.
    - Consistency–Constraints and data types are enforced during the execution of each command guaranteeing consistency.
    - Isolation–Commands run one at a time without overlap, providing for serializable isolation.
    - Durability–If all of the commands have deterministic side-effects (running the same queue of commands in the same order against the same starting dataset is promised to produce the same ending dataset), then writing (and fsync-ing) the command queue to disk before executing the commands makes all data changes durable.

No other stream processing solution on the market today can provide the same stateful stream processing capabilities: speed, scalability, and fault tolerance in one integration platform. In addition to contextual state, VoltDB also features capabilities essential to complex Event Stream Processing (ESP), such as:

- **Complex Event Processing (CEP) with Complex Data Models:** Data streams are getting more complicated with the dawn of IoT. VoltDB has a history of performing CEP on data streams: correlating multiple event sources, detecting complex patterns across these sources, enriching new data streams along with enriching historical data. Unlike other stream processing solutions on the market today VoltDB was built to handle complex data models, with the ability to:
  - Have one event affect multiple tables and streams
  - Provide comprehensive views for sophisticated analytics
  - Implement windowing functions with stream views
- **Smart(er) Stream Processing:** While, traditional stream processing platforms can do basic machine learning (ML) and pattern recognition, they lack the ability to:
  - Make complex decisions using hundreds to thousands of variables with contextual state in milliseconds.
  - Dynamically train and update ML models based on not just data from the stream, but also historical big data from a data lake/data warehouse.

In-memory NewSQL relational database management systems (RDBMS) were architected from the very first line of code for complex fast data processing. With RDBMS core functionality – such as User Defined Functions (UDF) and Stored Procedures – complex ML models customized for your business can be embedded in-platform for real-time actions/decisions on streaming data. PMML models are automatically converted into an executable process as a UDF and implemented in production. Only a NewSQL in-memory RDBMS can offer the necessary combination of complex actionable decisions + low latency + high throughput required for modern web scale apps.

- **Low latency Actions:** Analyzing streams in real-time is important, however, generating a low latency "action" from the analysis is critical for your business. With its in-memory distributed architecture, built-in stored procedures, and a host of other features, VoltDB has been proven to provide low latency actionable responses in the most demanding use cases.
- **Comprehensive *New* SQL:** SQL is a proven, long established standard to query data. Attempts to replace SQL with either MapReduce in the Apache Hadoop framework – or one of the many NoSQL tools – have failed miserably. Ironically, many "NoSQL" tools have pivoted back to adding SQL or "SQL-like" query languages. Apache Kafka jumped on the SQL bandwagon with "KSQL" for stream processing. KSQL, however, is far from standard SQL.

  VoltDB on the other hand was born as a NewSQL RDBMS; it offers fully ANSI compliant SQL on streaming data, enabling a much wider range of queries and complex event processing. Additionally, ANSI compliant NewSQL provides developers with the familiarity, flexibility and standardization necessary to build apps that rely on fast stream processing of data. NewSQL provides the same scalable performance of NoSQL systems for Hybrid Transactional & Analytical Processing (HTAP) workloads without compromising on ACID guarantees.

Stateful Stream Processing Enables NEW Use Cases

IoT, the shift to the cloud, along with the increased number of hyper connected devices, sensors powered by the faster connectivity of 5G are driving the complexity and volume of streaming data.

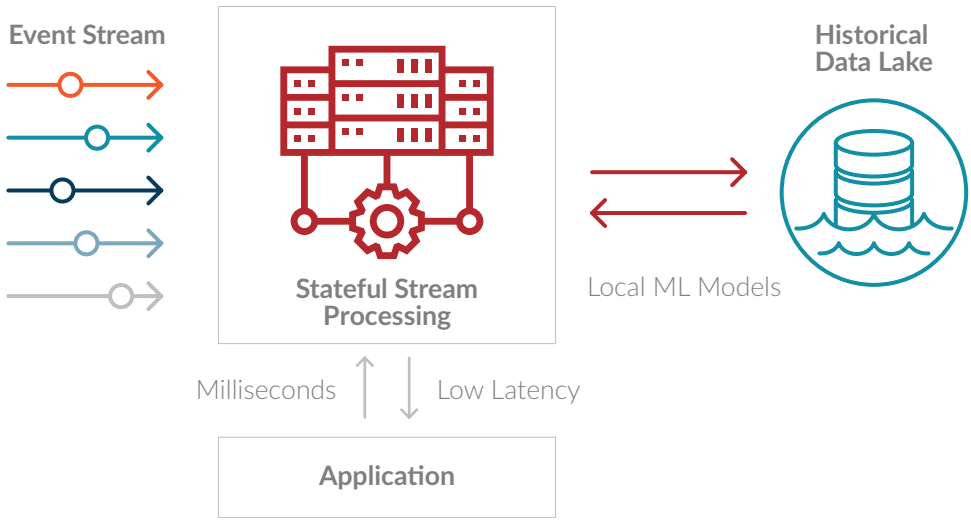
Next generation apps in the IoT era rely on:

1. Information derived from multiple sources/streams–Complex Event Processing (CEP)
2. Processing a continuous stream of data in real-time–at low latency
3. Take intelligent actions to drive desirable business outcomes–with embedded Machine Learning

Traditional stream processing platforms were originally architected as pub/sub message bus technologies; these are incapable of stateful, intelligent CEP, at low-latency. In this section, we will describe and provide examples of use cases which rely on these attributes for success.

Responsive Stream Processing

A lot of modern day use cases have an app that is awaiting a response from the analysis done by the up-stream Event Stream Processing system, as represented in Figure 1 below.



Event Stream Processing

ML models can be embedded as User Defined Functions (UDFs) in-platform for real-time intelligent decisioning. The Stateful Stream Processing engine enables developers to operationalize ML models built by data scientists in languages such as PMML. Complex PMML models are automatically converted into User Defined Functions (UDFs). The ML UDF then works like a standard SQL function. Embedded ML models can be trained and updated several times a day/as necessary based on new and historical data, to ensure they remain relevant to the business.

The app makes a request to the Stateful Stream Processing platform to update stored data, run queries and invoke the embedded ML models. The platform then provides the app with an intelligent response in real-time.

An example of a responsive stream processing use case in the Telecom industry is customer value management:

Many telecom operators are using alerts to upsell data blocks (e.g. 500MB for $5, valid until start of new billing period) and service passes (e.g. for roaming outside of the USA). A lot of operators have gone beyond this basic customer value management use case, and have started upselling real-time contextually aware offers. And as 5G gets rolled out across the globe, operators will need to develop exponentially more offers for an increasingly segmented customer base; the ability to be able to apply pricing and charging rules in real-time, to a wide range of new offers is fundamental.

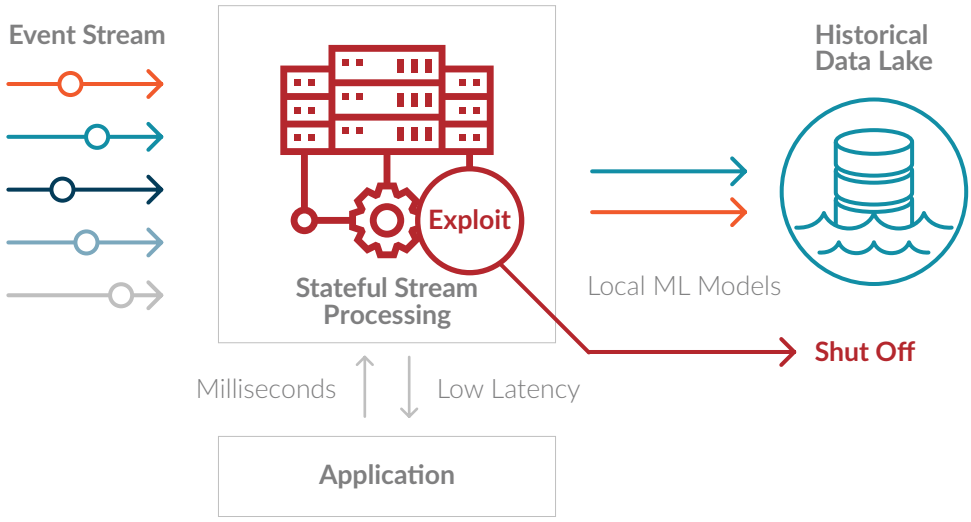
It starts by ingesting mobile phone usage events in real-time adn detect marketing opportunity events, such as:

- First time use of a specific feature
- Plan usage thresholds: 50%, 75%, 90%, 95%, 100%
- Time left, or time since activation, first use, etc
- Personalized offers by segment and history

Before the customer hangs up the phone, an offer is made to them to buy additional airtime.

Transitional Stream Processing

Transitional stream processing involves updating the stored state as multiple streams pass through the Stateful Stream Processing platform. The analysis performed by leveraging contextual state, along with the complex interactions of multiple streams, results in the generation of a new stream which is utilized to take a desired action downstream. Figure 2 provides a representative architecture of a typical transitional stream processing use case.

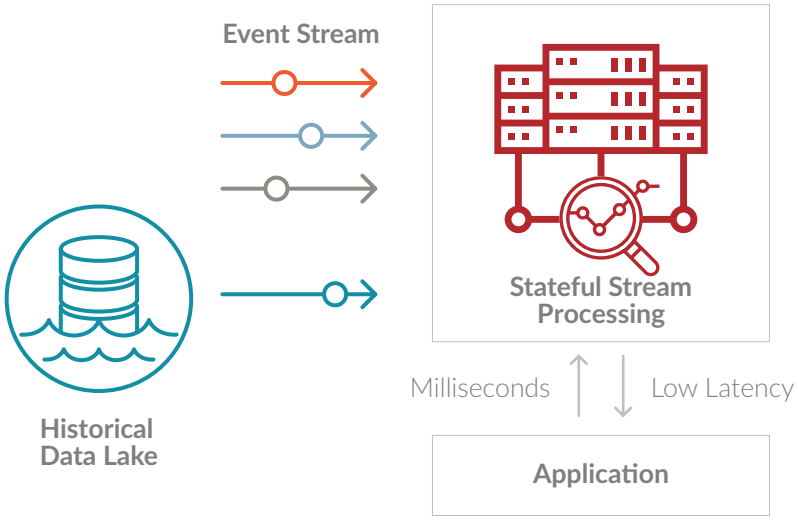


Transitional Stream Processing

The eCommerce/Website clickstream analytics use case falls under the transitional stream processing category. Analyzing clickstream data streams (the series of clicks a site visitor makes on your website) provides rich insight into which pages are effective and which pages site visitors ignore. When analyzed along with sales and conversion data streams, clickstream analysis can help discover the most effective series of steps needed to encourage conversions, sales, and add-on purchases. For example, after a customer adds a specific golf club to their cart, based on cohort analysis they would instantly get a recommendation for golf shoes, balls, and tees that they could also add to their cart.

Convergent Stream Processing

Convergent stream processing use cases ingest data from multiple event streams and potentially historical data from a data warehouse/data lake. While the app queries and receives responses from the Stateful Processing Platform in this scenario, there is no exit stream. Figure 3 represents a typical Convergent Stream Processing architecture.



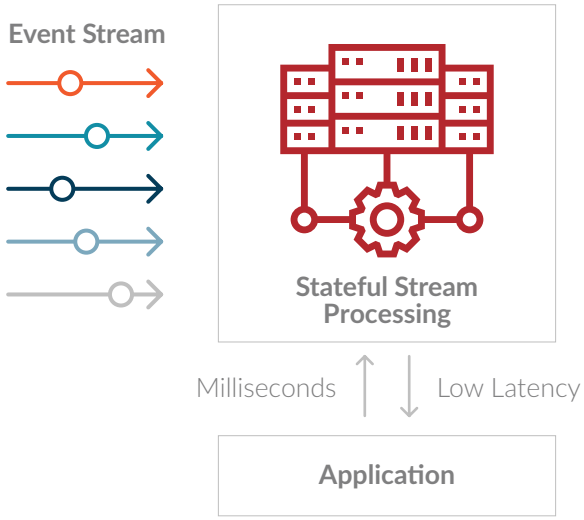
Convergent Stream Processing

Telecom fraud detection and prevention falls under this category. Wangiri (Japanese word for “one ring”) fraud consists of fraudsters calling thousands of unsuspecting subscribers and hanging up after one ring. If the subscribe calls back the missed call number, they are charged high international call termination rates. Post event analysis does not help with Wangiri fraud, as the subscriber is already billed for the high international rates, subscribers are unhappy with the telecom operator, and the brand reputation of the operator is tarnished.

Analyzing streaming data to detect anomalies in call attributes, such as: format, geolocation, blacklisted numbers, etc enables real-time detection of fraudulent calls. The Stateful Stream processing platform can identify fraudulent calls and block them before they terminate through a responsive action.

Hybrid Stream Processing

Use cases that fall under the hybrid stream processing category are a combination of either: 1) Responsive and Transitional stream processing, or 2) Responsive and Convergent stream processing.



Hybrid Stream Processing

Policy and Charging Function (PCRF), which is a part of Operations Support System (OSS) and Billing Support System (BSS) applications in the Telecom industry, is an excellent example of Hybrid Stream processing.

Historically, vendors have their own policies that were hardwired into the network. Here are some examples of functionality that is typically hardwired into the solution provider's hardware:

- Switching of video vs non-video content
- Prioritizing around a congested cell

With the emergence of Software Defined Networking (SDN) and Network Function Virtualization (NFV), switches and routers are no longer hardware-driven. They are software-driven and the rules are not hardcoded on the box, but the equipment calls into a common policy repository.

Stream processing solutions on the market today were originally architected as message queues. Their inherent architecture limits them from offering transactional and analytical processing capabilities.

A Stateful Stream Processing engine, such as VoltDB, provides just that. It stores subscriber information and attributes need to be accessed quickly, along with session data. With a stateful in-memory system, Telecom solution providers can prioritize connecting calls within the same cell ahead of calls across different cells; a hospital device sensor can be prioritized ahead of a gas meter sensor for IoT networks.

The VoltDB powered PCRF system is also responsible for:

- Billing: Applying a policy for post-paid subscribers where the bill gets added for every minute of air-time or data being used based on different attributes. For instance, who the subscriber is, what plan they have and where are they located, etc.
- Charging: Similar to billing, but for prepaid subscribers where the amount gets deducted to the prepaid minutes they may already have.
- Regulatory Compliance: Regulations require customers to get real-time alerts when they near their data quota limits and when they are roaming.

Summary

VoltDB's Stateful Stream processing solution is perfect for modern apps that require real-time complex actionability. Embedded contextual state persistence is a crucial requirement for low latency decisioning; no other solution on the market today offers a local state machine. Other tools require either writing code to maintain event state or writing glue code to connect to a disparate database that provides state. Both approaches add on latency and are expensive to build and maintain.

VoltDB's Stateful Stream processing solution provides state, CEP with complex data models, fault tolerance, strong ACID semantics, along with the capability of familiar SQL queries, all out-of-the-box.

Stateful Stream Processing enables a host of low latency responsive use cases that are just not possible with traditional Event Stream Processing technology.

Stateful Stream Processing enables a host of low latency responsive use cases that are just not possible with traditional Event Stream Processing technology.

About VoltDB

VoltDB powers applications that require real-time intelligent decisions on streaming data for a connected world, without compromising on ACID requirements. No other database can fuel applications that require a combination of speed, scale, volume and accuracy.

Architected by the 2014 A.M. Turing Award winner, Dr. Mike Stonebraker, VoltDB is a ground-up redesign of the relational database for today's growing real-time operations and machine learning challenges. Dr. Stonebraker has conducted research on database technologies for more than 40 years, leading to numerous innovations in fast data, streaming data and in-memory databases. With VoltDB, he realized the full potential of tapping streaming data with in-memory transactional database technology that can handle data's speed and volume while delivering real-time analytics and decision making. VoltDB is a trusted name in the industry already validated by leading organizations like: Nokia, Financial Times, Mitsubishi Electric, HPE, Barclays, Huawei, and more.

9 September 2019

VOLTDB

 209 Burlington Road, Suite 203, Bedford, MA 01730 voltdb.com