Real-Time Stream Processing as Game Changer in a Big Data World with Hadoop and Data Warehouse

The demand for stream processing is increasing a lot these days. The reason is that often processing big volumes of data is not enough.

Data has to be processed fast, so that a firm can react to changing business conditions in real time.

This is required for trading, fraud detection, system monitoring, and many other examples.

A “too late architecture” cannot realize these use cases.

This article discusses what stream processing is, how it fits into a big data architecture with Hadoop and a data warehouse (DWH), when stream processing makes sense, and what technologies and products you can choose from

Big Data versus Fast Data

Big data is one of the most used buzzwords at the moment. You can best define it by thinking of three Vs: Big data is not just about **V**olume, but also about **V**elocity and **V**ariety (see figure 1).

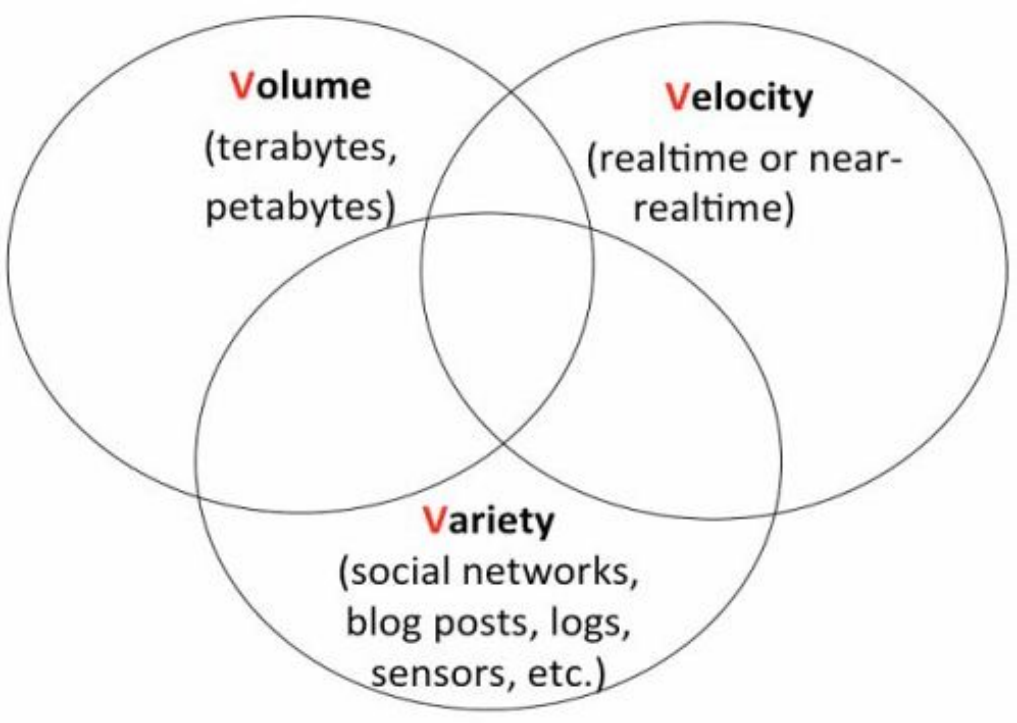

Figure 1: The three Vs of Big Data

A big data architecture contains several parts. Often, masses of structured and semi-structured historical data are stored in Hadoop (Volume + Variety). On the other side, stream processing is used for fast data requirements (Velocity + Variety). Both complement each other very well. This article focuses on real-time and stream processing. The end of the article discusses how to combine real-time stream processing with data stores such as a DWH or

Hadoop.

Having defined big data and its different architectural options, the next section explains what stream processing actually means.

The Definition of Stream Processing and Streaming Analytics

“Streaming processing” is the ideal platform to process data streams or sensor data (usually a high ratio of event throughput versus numbers of queries), whereas “complex event processing” (CEP) utilizes event-by-event processing and aggregation (e.g. on potentially out-of-order events from a variety of sources – often with large numbers of rules or business logic). CEP engines are optimized to process discreet “business events” for example, to compare out-of-order or out-of-stream events, applying decisions and reactions to event patterns, and so on. For this reason multiple types of event processing have evolved, described as queries, rules and procedural approaches (to event pattern detection). The focus of this article is on stream processing.

Stream processing is designed to analyze and act on real-time streaming data, using “continuous queries” (i.e. SQL-type queries that operate over time and buffer windows). Essential to stream processing is Streaming Analytics, or the ability to continuously calculate mathematical or statistical analytics on the fly within the stream. Stream processing solutions are designed to handle high volume in real time with a scalable, highly available and fault tolerant architecture. This enables analysis of data in motion.

In contrast to the traditional database model where data is first stored and indexed and then subsequently processed by queries, stream processing takes the inbound data while it is in flight, as it streams through the server. Stream processing also connects to external data sources, enabling applications to incorporate selected data into the application flow, or to update an external database with processed information.

A recent development in the stream processing industry is the invention of the “live data mart” which provides end-user, ad-hoc continuous query access to this streaming data that’s aggregated in memory. Business user-oriented analytics tools access the data mart for a continuously live view of streaming data. A live analytics front ends slices, dices, and aggregates data dynamically in response to business users’ actions, and all in real time.

Figure 2 shows the architecture of a stream processing solution, and the live data mart.

[Click on the image to enlarge it]

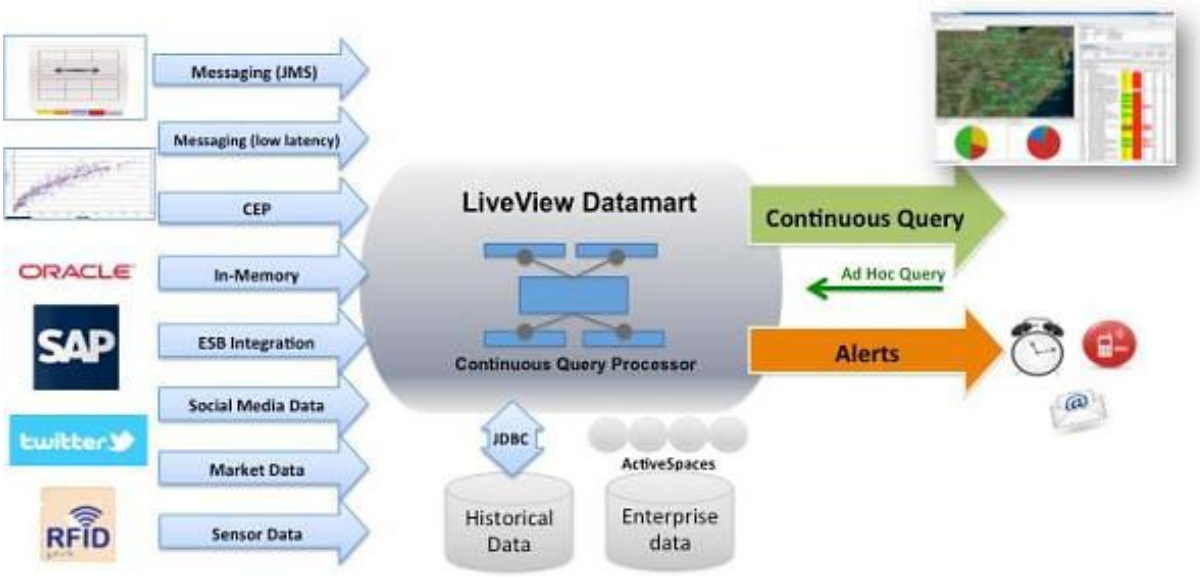

Figure 2: Stream Processing Architecture

A stream processing solution has to solve different challenges:

- Processing massive amounts of streaming events (filter, aggregate, rule, automate, predict, act, monitor, alert)
- Real-time responsiveness to changing market conditions
- Performance and scalability as data volumes increase in size and complexity
- Rapid integration with existing infrastructure and data sources: Input (e.g. market data, user inputs, files, history data from a DWH) and output (e.g. trades, email alerts, dashboards, automated reactions)
- Fast time-to-market for application development and deployment due to quickly changing landscape and requirements
- Developer productivity throughout all stages of the application development lifecycle by offering good tool support and agile development
- Analytics: Live data discovery and monitoring, continuous query processing, automated alerts and reactions
- Community (component / connector exchange, education / discussion, training / certification)
- End-user ad-hoc continuous query access
- Alerting
- Push-based visualization

Now I've explained what stream processing is, the next section will discuss some use cases where an enterprise needs stream processing to get positive business outcomes.

Real World Stream Processing Use Cases

Stream processing found its first uses in the finance industry, as stock exchanges moved from floor-based trading to electronic trading. Today, it makes sense in almost every industry - anywhere where you generate stream data through human activities, machine data or sensors data. Assuming it takes off, the Internet of Things will increase volume, variety and velocity of data, leading to a dramatic increase in the applications for stream processing technologies. Some use cases where stream processing can solve business problems include:

- Network monitoring
- Intelligence and surveillance
- Risk management
- E-commerce
- Fraud detection
- Smart order routing
- Transaction cost analysis
- Pricing and analytics
- Market data management
- Algorithmic trading
- Data warehouse augmentation

Let's discuss one use case in more detail using a real world example.

Real-Time Fraud Detection

This fraud detection use case is from one of my employer's clients in the finance sector, but it is relevant for most verticals (the actual fraud event analytics and data sources differ among different fraud scenarios). The company needs to monitor machine-driven algorithms, and look for suspicious patterns. In this case, the patterns of interest required correlation of five streams of real-time data. Patterns happen within 15-30 second windows, during which thousands of dollars could be lost. Attacks come in bursts. Previously, the data required to find these patterns was loaded into a DWH and reports were checked each day. Decisions to act were made every day. But new regulations ir the capital markets require firms to understand trading patterns in real time, so the previous DWH-based architecture is now "too late" to comply with trading regulations.

The stream processing implementation now intercepts the data before it hits the DWH by connecting StreamBase directly to the source of trading.

Mark Palmer takes up the story in more detail:

> *Once this firm could see patterns of fraud, they were faced with a new challenge: What to do about it? How many times did the pattern need to be repeated until active surveillance is started? Should the action be quarantined for a period, or halted immediately? All these questions were new, and the answer to them keeps changing.*
>
> *The fact that the answers keep changing highlights the importance of ease of use. Analytics must be changed quickly and be made available to fraud experts - in some cases, in hours - as understanding deepens, and as the bad guys change their tactics.*

The end of the article will describe some more real world use cases, which combine stream processing with a DWH and Hadoop.

Comparison of Stream Processing Alternatives

Stream processing can be implemented by doing-it-yourself, using a framework or a product. Doing-it-yourself should not be an option in most cases, because there are good open source frameworks available for free. However, a stream processing product might solve many of your issues out-of-the-box, while a framework still requires a lot of self-coding and the Total Cost of Ownership might be much higher than expected compared to a product.

From a technical perspective, the following components are required to solve the described challenges and implement a stream processing use case:

- **Server**: An ultra-low-latency application server optimized for processing real-time streaming event data at high throughputs and low latency (usually in-memory).
- **IDE**: A development environment, which ideally offers visual development, debugging and testing of stream processing processes using streaming operators for filtering, aggregation, correlation, time windows, transformation, etc. Extendibility, e.g. integration of libraries or building custom operators and connectors, is also important.
- **Connectors**: Pre-built data connectivity to communicate with data sources such as database (e.g. MySQL, Oracle, IBM DB2), DWH (e.g. HP Vertica), market data (e.g. Bloomberg, FIX, Reuters), statistics (e.g. R, MATLAB, TERR) or technology (e.g. JMS, Hadoop, Java, .NET).
- **Streaming Analytics**: A user interface, which allows monitoring, management and real-time analytics for live streaming data. Automated alerts and human reactions should also be possible.

- **Live Data Mart and/or Operational Business Intelligence**: Aggregates streaming data for ad-hoc, end-user, query access, alerting, dynamic aggregation, and user management. Live stream visualization, graphing, charting, slice and dice are also important.

As of end-2014, only a few products are available on the market that offer these components. Often, a lot of custom coding is required instead of using a full product for stream processing. The following gives an overview about well-known and widely adopted options.

Apache Storm

Apache Storm is an open source framework that provides massively scalable event collection. Storm was created by Twitter and is composed of other open source components, especially ZooKeeper for cluster management, ZeroMQ for multicast messaging, and Kafka for queued messaging.

Storm runs in production in several deployments. Storm is in the incubator stage of Apache's standard process - current version is 0.9.1-incubating. No commercial support is available today, though Storm is adopted more and more. In the meantime, some Hadoop vendors such as Hortonworks are adding it to their platform step by step. The current release of Apache Storm is a sound choice if you are looking for a stream processing framework. If your team wants to implement a custom application by coding without any license fees, then Storm is worth considering. Brian Bulkowski, founder of Aerospike (a company which offers a NoSQL database with connectors to Storm) has great introductory slides, which let you get a feeling about how to install, develop and run Storm applications. Storm's website shows some reference use cases for stream processing at companies such as Groupon, Twitter, Spotify, HolidayCheck, Alibaba, and others.

Apache Spark

Apache Spark is a general framework for large-scale data processing that supports lots of different programming languages and concepts such as MapReduce, in-memory processing, stream processing, graph processing or machine learning. This can also be used on top of Hadoop. Databricks is a young startup offering commercial support for Spark. Hadoop distributors Cloudera and MapR partner with Databricks to offer support. As Spark is a very young project, only a few reference use cases are available yet. Yahoo uses Spark for personalizing news pages for web visitors and for running analytics for advertising. Conviva uses Spark Streaming to learn network conditions in real time.

IBM InfoSphere Streams

InfoSphere Streams is IBM's flagship product for stream processing. It offers a highly scalable event server, integration capabilities, and other typical features required for implementing stream processing use cases. The IDE is based on Eclipse and offers visual development and configuration (see figure 5: IBM InfoSphere Streams IDE).

[Click on the image to enlarge it]

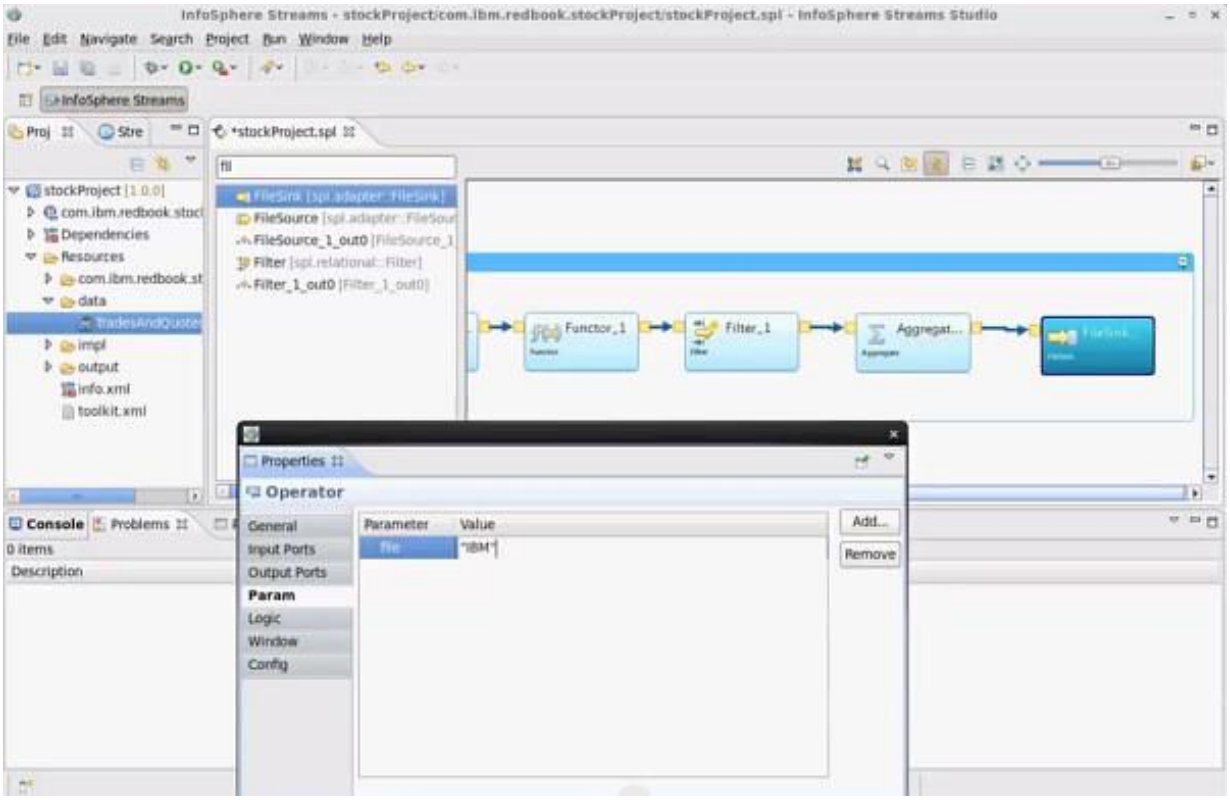

Figure 3: IBM InfoSphere Streams IDE

Zubair Nabi and Eric Bouillet from IBM Research Dublin, along with Andrew Bainbridge and Chris Thomas from IBM Software Group Europe, created a benchmark study, (pdf) which gives some detailed insights about IBM InfoSphere Streams and compares it to Apache Storm. Amongst other things their study suggests that InfoSphere Streams significantly outperforms Storm.

TIBCO StreamBase

TIBCO StreamBase is a high-performance system for rapidly building applications that analyze and act on real-time streaming data. The goal of StreamBase is to offer a product that supports developers in rapidly building real-time systems and deploying them easily (see figure 3: TIBCO StreamBase IDE).

[Click on the image to enlarge it]

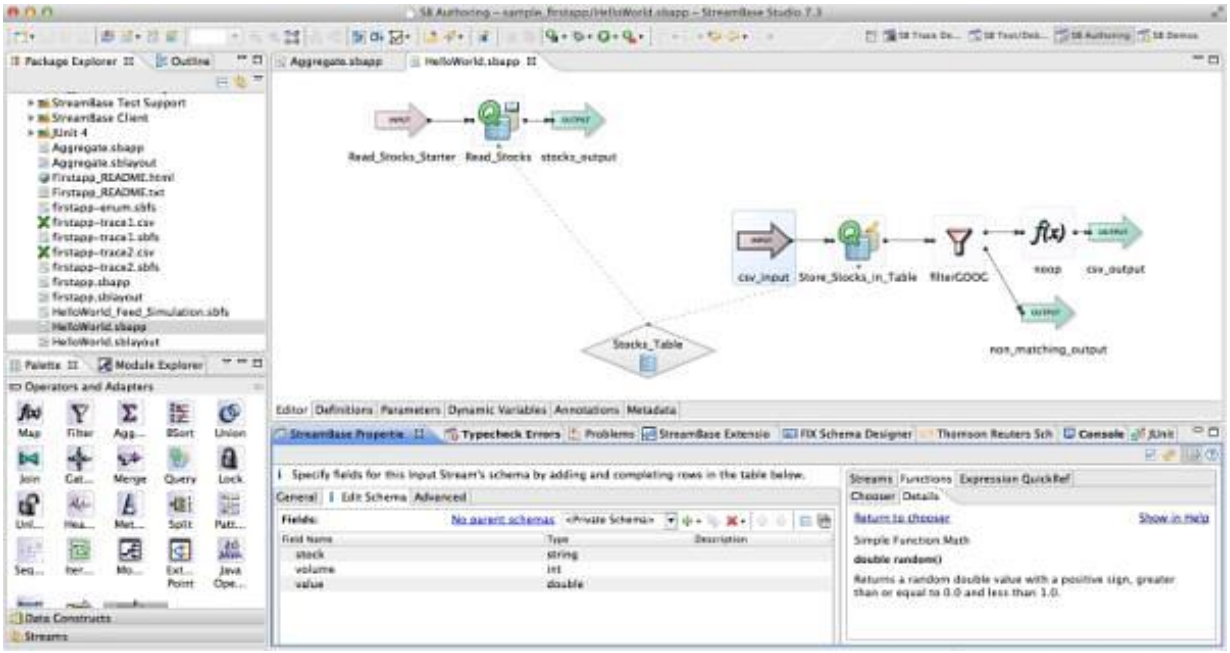

Figure 4: TIBCO StreamBase IDE

StreamBase LiveView data mart is a continuously live data mart that consumes data from streaming real-time data sources, creates an in-memory data warehouse, and provides push-based query results and alerts to end users (see figure 4: TIBCO StreamBase LiveView). At the time of writing, no other vendor offers a live data mart for streaming data.

[Click on the image to enlarge it]

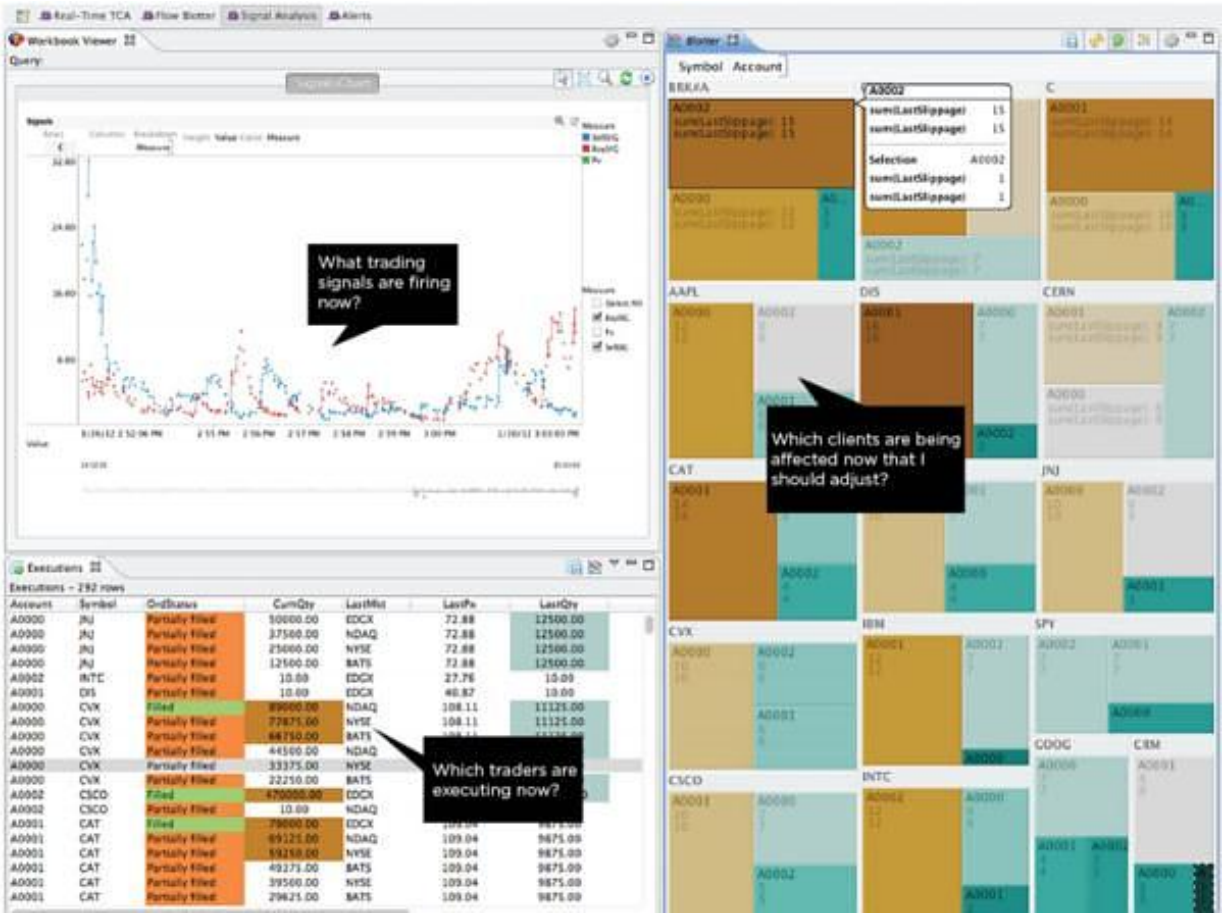

Figure 5: TIBCO StreamBase LiveView

The StreamBase LiveView Desktop is a push-based application that communicates with the server, the live data mart. The desktop allows business users to analyze, anticipate and act on streaming data. It supports end-user alert management and interactive action on all visual elements in the application. In the desktop the end user can spot a real-time condition that appears to be fraud, click on the element on the screen, and stop the trading order in real time. In this way, the desktop is not just a passive “dashboard”, but also an interactive command and control application for business users. There are several commercial desktop-only dashboard offerings, such as Datawatch Panopticon. It should be noted however that most dashboard products are designed for passive data viewing, rather than interactive action.

Other Stream Processing Frameworks and Products

Some other open source frameworks and proprietary products are available on the market. The following is a short overview (this is not a complete list).

Open Source:

Proprietary:

- AWS Kinesis: A managed cloud service from Amazon for real-time processing of streaming data. It is deeply integrated with other AWS cloud services such as S3, Redshift or DynamoDB.
- DataTorrent: A real-time streaming platform that runs natively on Hadoop.
- Most big software vendors also offer some kind of stream processing within their Complex Event Processing (CEP) products, e.g. Apama from Software AG, Oracle CEP or SAP’s Sybase CEP.

Most frameworks and products sound very similar when you read the websites of the vendors. All offer real-time stream processing, high scalability, great tools, and awesome monitoring. You definitely have to try them out before buying (if they will let you) to see the differences for yourself regarding ease of use, rapid development, debugging and testing, real-time analytics, monitoring, etc.

Evaluation: Choose a Stream Processing Framework or a Product or Both?

The typical evaluation process (long list, short list, proof of concept) is obligatory before making a decision.

Compared to frameworks such as Apache Storm or Spark, products such as IBM InfoSphere Streams or TIBCO StreamBase differentiate with:

- A stream processing programming language for streaming analytics
- Visual development and debugging instead of coding
- Real-time analytics
- Monitoring and alerts
- Support for fault tolerance, and highly optimized performance
- Product maturity
- In the case of TIBCO, a live data mart and operational command and control center for business users
- Out-of-the-box connectivity to plenty of streaming data sources
- Commercial support
- Professional services and training.

Think about which of the above features you need for your project. In addition, you have to evaluate costs of using a framework against productivity, reduced effort and time-to-market using a product before making your choice.

Because of the gaps (language, tooling, data mart, etc.) in Apache Storm, it is sometimes used in conjunction with a commercial stream processing platform. So, stream processing products can be complementary to Apache Storm. If Storm is already used in production for collecting and counting streaming data, a product can leverage its advantages to help with integrating other external data sources and analyzing, querying, visualizing, and acting on combined data, e.g. by adding visual analytics easily without coding. Some companies already use this architecture shown in figure 6. Such a combination also makes sense for other stream processing solutions such as Amazon's Kinesis.

[Click on the image to enlarge it]

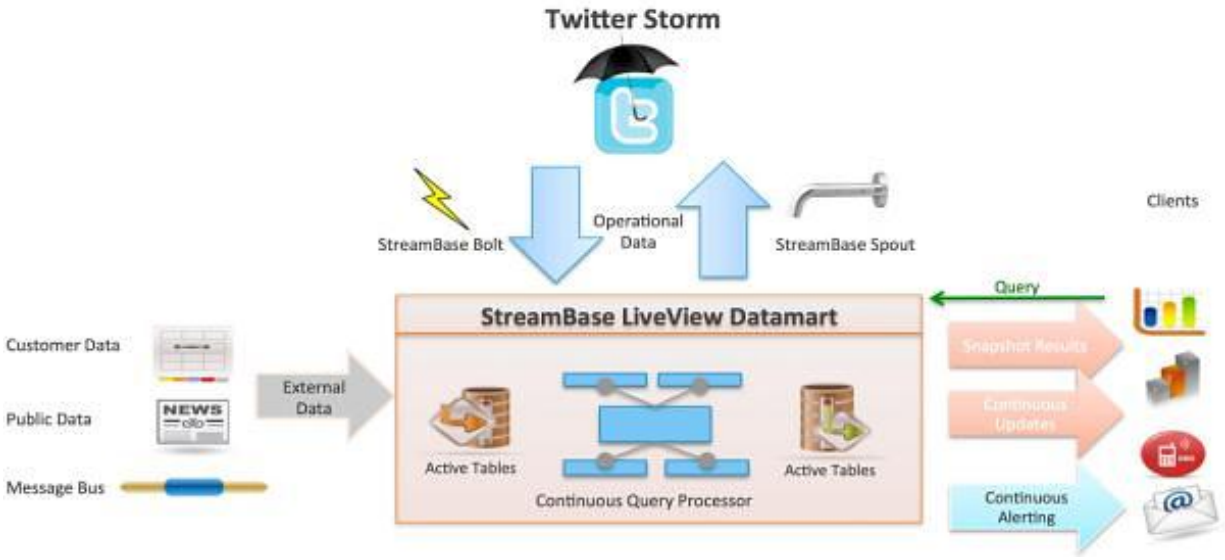

Figure 6: Combination of a Stream Processing Framework (for Collection) and Product (for Integration of External Data and Streaming Analytics)

Besides evaluating the core features of stream processing products, you also have to check integration with other products. Can a product work together with messaging, Enterprise Service Bus (ESB), Master Data Management (MDM), in-memory stores, etc. in a loosely coupled, but highly integrated way? If not, there will be a lot of integration time and high costs.

Having discussed different frameworks and product alternatives, let's take a look at how stream processing fits into a big data architecture. Why and how to combine stream processing with a DWH or Hadoop is described in the next section.

Relation of Stream Processing to Data Warehouse and Hadoop

A big data architecture contains stream processing for real-time analytics and Hadoop for storing all kinds of data and long-running computations. A third part is the data warehouse (DWH), which stores just structured data for reporting and dashboards. See "Hadoop and DWH – Friends, Enemies or Profiteers? What about Real Time?" for more details about combining these three parts within a big data architecture. In summary, big data is not just Hadoop; concentrate on business value! So the question is not an "either / or" decision. DWH, Hadoop and stream processing complement each other very well. Therefore, the integration layer is even more important in the big data era, because you have to combine more and more different sinks and sources.

Stream Processing and DWH

A DWH is a great tool to store and analyze structured data. You can store terabytes of data and get answers to your queries about historical data within seconds. DWH products such as Teradata or HP Vertica were built for this use case. However the ETL processes often take too long. Business wants to query up-to-date information instead of using an approach where you may only get information about what happened yesterday. This is where stream processing comes in and feeds all new data into the DWH immediately. Some vendors already offer this combination. For example, Amazon's cloud offering includes Amazon Kinesis for real-time stream processing and connectors to its DWH solution Amazon Redshift.

A real world use case of this is at BlueCrest (one of Europe's leading hedge funds), which combines HP Vertica as DWH and TIBCO StreamBase to solve exactly this business problem. BlueCrest uses StreamBase as a real-time pre-processor of market data from disparate sources into a normalized, cleansed, and value-added historical tick store. Then complex event processing and the DWH are used as data sources to their actual trading systems using StreamBase's connectors.

Another set of use cases are around using stream processing as a "live data mart" using that to front-end both streaming data and a historical store in a DWH through a unified framework. TIBCO LiveView is an example for building such a "live data mart" easily. Besides acting automatically, the "live data mart" offers monitoring and operations in real time to humans.

IBM also describes some interesting use cases for DWH modernization using Stream Processing and Hadoop capabilities:

- Pre-Processing: Using big data capabilities as a "landing zone" before determining what data should be moved to the data warehouse.
- Offloading: Moving infrequently accessed data from DWHs into enterprise-grade Hadoop.
- Exploration: Using big data capabilities to explore and discover new high value data from massive amounts of raw data and free up the DWH for more structured, deep analytics.

Stream Processing and Hadoop

A combination of stream processing and Hadoop is key for IT and business. Hadoop was never built for real-time processing.

Hadoop initially started with MapReduce, which offers batch processing where queries take hours, minutes or at best seconds. This is and will be great for complex transformations and computations of big data volumes. However, it is not so good for ad hoc data exploration and real-time analytics. Multiple vendors have though made improvements and added capabilities to Hadoop that make it capable of being more than just a batch framework. For example:

- Hive Stinger Initiative from Hortonworks to improve and accelerate SQL queries with MapReduce jobs.
- New query engines, e.g. Impala from Cloudera or Apache Drill from MapR, which do not use MapReduce at all.
- DWH vendors, e.g. Teradata, EMC Greenplum, combine Hadoop with their DWH and add their own SQL query engines, again without MapReduce under the hood.
- Summingbird, created and open sourced by Twitter, enables developers to uniformly execute code in either batch-mode (Hadoop/MapReduce-based) or stream-mode (Storm-based), so both concepts can be combined within a single framework - for more details see this news.

Storm and Spark were not invented to run on Hadoop, but now they are integrated and supported by the most popular Hadoop distributions (Cloudera, Hortonworks, MapR), and can be used for implementing stream processing on top of Hadoop. The lack of maturity and good tooling are obstacles you usually have to live with with early open source tools

and integrations, but you can get a lot done and these are great learning tools. Some stream processing products developed connectors (using Apache Flume in the case of StreamBase) to Hadoop, Storm, etc., and might therefore be a good alternative to a framework for combining stream processing and Hadoop.

Let's take a look at a real world use case for this combination of stream processing and Hadoop. TXODDS offers real-time odds aggregation for the fast-paced global sports betting market. TXODDS selected TIBCO StreamBase for zero-latency analytics in combination with Hadoop. The business scenario is that 80 percent of betting takes place after the actual sporting event has started, and that TXODDS needs to better anticipate and predict pricing movements. Intelligent decisions must be made on thousands of concurrent games and in real time. Using just ETL and batch processing to compute odds before a match starts are not enough any more.

The architecture of TXODDS has two components. Hadoop stores all history information about all past bets. MapReduce is used to pre-compute odds for new matches, based on historical data. StreamBase computes new odds in real time to react within a live game after events occur (e.g. when a team scores a goal or a player gets sent off). Historical data from Hadoop is also brought into this real-time context. In this video, Alex Kozlenkov, Chief Architect at TXODDS discusses the technical architecture in detail.

Another great example is PeerIndex, a startup providing social media analytics based on footprints from the use of major social media services (currently Twitter, LinkedIn, Facebook and Quora). The company delivers influence at scale by exposing services built on top of their influence graph; a directed graph of who is influencing whom on the web.

PeerIndex gathers data from the social networks to create the influence graph. Like many startups, they use a lot of open source frameworks (Apache Storm, Hadoop, Hive) and elastic cloud infrastructure services (AWS S3, DynamoDB) to get started without spending much money on licenses, but yet still be able to scale quickly. Storm processes their social data, to provide real-time aggregations and to crawl the web, before storing the data in a manner most suitable for their Hadoop-based systems to do further batch processing.

Conclusion

Stream processing is required when data has to be processed fast and / or continuously, i.e. reactions have to be computed and initiated in real time. This requirement is coming more and more into every vertical. Many different frameworks and products are available on the market already, however the number of mature solutions with good tools and commercial support is small today. Apache Storm is a good, open source framework; however custom coding is required due to a lack of development tools and there's no commercial support right now. Products such as IBM InfoSphere Streams or TIBCO StreamBase offer complete products, which close this gap. You definitely have to try out the different products, as the websites do not show you how they differ regarding ease of use, rapid development and debugging, and real-time streaming analytics and monitoring. Stream processing complements other technologies such as a DWH and Hadoop in a big data architecture - this is not an "either/or" question. Stream processing has a great future and will become very important for most companies. Big Data and Internet of Things are huge drivers of change.

About the Author

Kai Wähner works as Technical Lead at TIBCO. All opinions are his own and do not necessarily represent his employer. Kai's main area of expertise lies within the fields of Application Integration, Big Data, SOA, BPM, Cloud Computing, Java EE and Enterprise Architecture Management. He is speaker at international IT conferences such as JavaOne, ApacheCon, JAX or OOP, writes articles for professional journals, and shares his experiences with new technologies on his blog. Contact: kontakt@kai-waehner.de or Twitter: @KaiWaehner. Find more details and references (presentations, articles, blog posts) on his website.