Big Data Analytics for Real Time Systems

BBSB ENGINEERING COLLEGE
PRINCE KUMAR (2001308)
PRAVIN KUMAR(2001305)

ABSTRACT

The digitization of virtually everything now creates new types of large and real-time data across a broad range of industries. Much of this is non-standard data: for example, streaming, geospatial or sensor-generated data that does not fit neatly into traditional, structured, relational warehouses. Today's advanced analytics technologies and techniques enable organizations to extract insights from data with previously unachievable levels of sophistication, speed and accuracy. Real time analytics for Big Data is about the ability to make better decisions and take meaningful actions at the right time. We provide an overview of the Big Data Analytics for Real Time Systems and focus on its challenges, research trends and accuracy of results.

1. INTRODUCTION

From the earliest forms of writing to modern data centers, the human race has always gathered information. The rise in technology has led to the overflow of data, which requires more sophisticated data storage systems. Seventy years ago, this growth rate in the volume of data was termed as "Information Explosion". Technological developments from the invention of the printing press through automatic acquisition of data from space exploration have foisted the information explosion. Ever-growing numbers of warehouses of data, both hard-copy records and magnetic tapes, attested the need for somehow condensing the volume of information while preserving its content.

In 1967, B. A. Marron & P. A. D. de Maine's Automatic Data Compression was introduced to deal with information explosion [Marron], however three decades later in October 1997, the need to suppress growth of data beyond information explosion was necessary and the term "Big Data" was first used in the proceedings of the IEEE 8th conference on Visualization to describe this voluminous growth of data [Cox]. In this Cox et al, proposed a solution of out-of-core visualization (When a single data set that we wish to visualize is larger than the capacity of main memory) and remote-out-of core visualization (When a single data set is larger than the capacity of local memory and disk).

In this paper, we provide a flow of information from big data, its analytics, real time systems, existing technologies and tools, and finally the use cases of real time analytics. By doing this we wish to gain insight on how and why organizations should accelerate their business operations into real-time. A number of real-world use cases are now established in which real-time techniques help organizations grow.

2015, Aachen, Germany.

1.1 Big Data

The term Big Data as coined today, is used to describe the exponential growth and availability of structured and unstructured data. However, Big Data is not just about lots of data, it is a concept providing an opportunity to find new insight into existing data as well guidelines to capture and analyze future data.

In 2001, industry analyst Doug Laney articulated the now mainstream definition of big data as the three Vs of big data: volume, velocity and variety [META group].

Volume: A number of factors contribute to increasing the volume of Data. Information is becoming a tangible resource and is not being discarded. Thus, Transaction-based data stored through the years, unstructured data streaming in from social media, sensors and machine to machine data being collected etc., contribute to the increasing volume of data which is handled by purchasing additional online storage. Other methods like implementing tiered storage systems, outsourcing data management, profiling data sources are being adopted. In the past the storage of data was the main concern, but with decreasing storage costs other issues emerge such as how to determine relevance within large data volumes and how to use analytics to create value from relevant data.

Velocity: Data is streamed in at unprecedented speed and must be dealt with in a timely manner. Reacting quickly enough to deal with data velocity is a challenge for most organizations. News channels, radios and social media have changed how fast we receive the news. The data movement is now almost real time and the update window has reduced to fractions of the seconds.

Variety: Data today comes in all types of formats. Structured numeric data in traditional databases, Information created from line-of-business applications etc. and unstructured text documents, email, video, audio, stock ticker data and financial transactions. Managing, merging and governing different varieties of data is something many organizations still grapple with.

More recently, additional Vs have been proposed for addition to the model, including variability -- the increase in the range of values typical of a large data set -- and value, which addresses the need for valuation of enterprise data.

1.2 Big Data Analytics

Big data analytics is the process of examining large data sets to uncover hidden patterns, unknown correlations, market trends, customer preferences and other useful business information. The analytics of big data plays an important role for decision making in business and society as a whole. Accurate analyses leads to

more confident decision-making, which means greater operational efficiencies, cost reductions and reduced risk.

Big data can be analyzed with the software tools such as predictive analytics, data mining, text analytics and statistical analysis. Mainstream BI software and data visualization tools can also play a role in the analysis process. But the semi-structured and unstructured data may not fit well in traditional data warehouses based on relational databases. Furthermore, data warehouses may not be able to handle the processing demands posed by sets of big data that need to be updated frequently or even continually. As a result, many organizations looking to collect, process and analyze big data have turned to a newer class of technologies that includes Hadoop and related tools such as YARN, MapReduce, Spark, Hive and Pig as well as NoSQL database. Those technologies form the core of an open source software framework that supports the processing of large and diverse data sets across clustered systems. Big data analytics is often associated with cloud computing because the analysis of large data sets in real-time requires a platform like Hadoop to store large data sets across a distributed cluster and MapReduce to coordinate, combine and process data from multiple sources.

A number of barriers need to be handled to reach the benefits of Big Data Analytics [Russom, 2011]. Inadequate staffing and skills are the leading barriers to big data analytics. Other important barriers are a lack of business support and Problems with database software. But with the increasing popularity of Big Data a number new platforms based on cloud are being introduced. Given the open source nature of Hadoop, many users are experimenting with it to analyze Big Data making it one of the most important tools.

1.3 Analytics and 3V's

With the introduction of the mainstream definition of Big Data by Doug Laney in 2001, a number of big companies have tried and succeeded in analyzing the data.

For example, Facebook's strategy to present data over and over again within the same parameters was an important milestone to analyze enormous volumes of data. This drove technology innovations such as column databases, which are now widely used by other companies that face equally sizable stores of similar data items.

When it comes to variety and velocity, though, too many enterprises still face a big problem in their approach to big data analytics. The velocity challenge has been largely addressed through sophisticated indexing techniques and distributed data analytics that enable processing capacity to scale with increased data velocity, but most analyses today, however, are not that continuous. The refresh of most deployed analyses is latent, with intraday refresh (every few hours, hourly, or in real time) being rare. This puts analytics behind the times, as compared to reporting, where real-time refresh is the norm for some reporting types [Russom, 2011].

Real-time processing of streaming big data is therefore crucial to finding meaningful data and reacting to it. The march toward real time is affecting many enterprise applications types; no doubt, analytics will soon come closer to real time as part of this trend.

2. BACKGROUND

A real-time system is one that processes information and produces a response within a specified time, else risk severe consequences, sometimes including failure.

2.1 Real Time Systems

Real-time is a broad concept involving many types of processes, tools, and users. It does not have to be 'fast': the deadline may be days or weeks. For instance, the time period could be as short as that for a weather forecasting system or as long as 48 hours at the Texas Instruments European Program Information Centre (epic) for literature requests.

Real-time software solutions monitor and analyze business activities to give a wide range of users the real-time visibility they need to see a problem or opportunity, make a fully informed decision, and then act accordingly. Many use cases for real-time technology are being deployed today, in many industries to enhance customer behavior, evaluate sales performance, identify a new social media sentiment etc. and one such major application is in the context of Big Data.

2.2 Real Time Systems in Context of Big Data

The value of analyzing data in motion can be significant, depending on the type of business. As user organizations move deeper into embedding analytics to transform information into insight and then action, they also move deeper into real time. For time-sensitive processes like catching fraud, bank applications and security, analytics of real time systems is necessary and every second counts.

Rapidly increasing use of large-scale and location-aware social media and mobile applications are driving the need for scalable, real-time platforms that can handle streaming analysis and processing of massive amounts of data. Today, creating an analytics system for big data generally means collecting multiple technologies from various providers, and building the system yourself.

2.3 Challenges of Big Data Analytics and Real Time Systems

Big Data analytics for Real-Time systems pose a number of technical and organizational challenges.

The cost of the real-time technology, architecting a complex real time system and the management of current state of data are the major challenges. On the other hand as clearly explained in the Big Data and Analytics Hub, the market is too dynamic to predict, subscribers' preferences change rapidly and the competition offers more accelerator to this mix, so it is very hard to get the right segment. The campaign does not produce the anticipated returns and, often, it is even challenging to measure the effectiveness of the campaign. Therefore the only option left for the operator is to quickly tweak the segmentation criteria (using a very agile tool set), hit the market and experiment on what really works. This typically has to undergo multiple iterations and in an agile manner. This is the *spray and prays* approach and that is not the end of it all. With that automatic weapon in the armory, the operator goes happily spraying until he realizes that his customers have quickly developed serious campaign fatigue [Raja SP, 2014].

Furthermore, majority of the data handled in real time today is structured (or more specifically relational), followed by application logs and semi-structured data, but there is an anticipation for more real-time handling of social media data, Web logs and clickstreams, and unstructured data.

Most organizations today are not making use of the analytic opportunities of big data because they still focus on insights from

structured data via traditional tools for business intelligence (BI) and data warehousing (DW). One of the challenges is that traditional BI/DW tools were not designed for new data sources and data types such as exponentially growing volumes of unstructured and semi-structured data, especially from new sources such as machines, sensors, logs, and social media

2.4 Current Trends in Big Data Analytics for Real Time Systems

2.4.1 Business Intelligence

From the very beginning, executives and other business users of Business Intelligence (BI), executive information, and decision support systems were often disappointed to find that, after considerable technology investment, they would not be getting real-time data in their applications [Russom et al., 2014]. Traditionally, BI applications have served up historical data that could be days, weeks, or even months old. Many organizations are also deploying dedicated analytic platforms, data accelerators, data appliances, and cloud-based solutions to speed up BI performance and take pressure off of existing systems. These technologies may still not deliver true real-time data to BI users, but they are enabling organizations to update or refresh data far more frequently and deliver answers to queries sooner.

Furthermore, there is an advancement of technologies like Hadoop, i.e., both open source and commercial applications for BI-style interactive querying and streaming-data analysis. As YARN-based technologies mature, organizations are able to realize greater value from Hadoop resources and integrate BI and analytic application access to them alongside other resources, such as enterprise data warehouses.

2.4.2 Operational Intelligence

Operational intelligence (OI) is an emerging class of analytics that provides visibility into business processes, events, and operations as they are happening. OI can handle machine data, sensor data, event streams, and other forms of streaming data and big data. OI solutions can also correlate and analyze data collected from multiple sources in various latencies (from batch to real time) to reveal actionable information.[Russom SPLUNK] Organizations can act on the information by immediately sending an alert to the appropriate manager, updating a management dashboard, offering an incentive to a churning customer, adjusting machinery, or preventing fraud. In the Use Cases section we further discuss the applications of OI.

3. TECHNOLOGIES

Big Data analytics for Real-Time systems pose a number of technical and organizational challenges.There is an on-going shift in data processing from the batch processing based to the real-time processing. A lot of technologies can be used to create a real-time processing system. It is complicated to choose the right tools, incorporate, and orchestrate them. Marz proposes a generic solution to this problem, called *Lambda Architecture* [Marz]. Figure 1 describes the lambda architecture. The Lambda Architecture proposed by Nathan Marz takes a very unique approach and solves the problem of computing arbitrary functions on a big data set in real-time by decomposing the problem into three layers: the batch layer, the serving layer, and the speed layer. The architecture consists of three layers. First, the batch layer computes views on the collected data, and repeats the process when it is done to infinity. Its output is always outdated by the time it is available for new data has been received in the meantime. Second, a parallel speed-processing layer closes this gap by constantly processing the most recent data in near real-time fashion. Any query against the data is answered through the serving layer, i.e., by querying both the speed and the batch layers' serving stores in the serving layer, and the results are merged to answer user queries. Speed layer is implemented by Storm (Trident), which computes ad-hoc functions on a data stream (time series facts) in real-time. The result from the incremental changes confined to a sliding window is then merged with the materialized batch view from the serving layer to generate up-to-date analysis results.

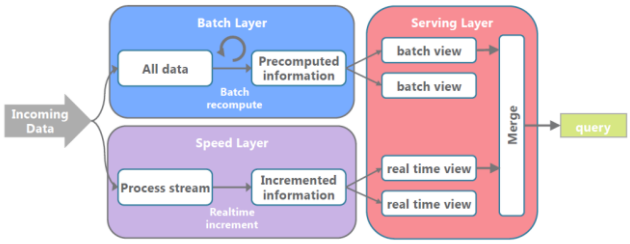

Figure 1. Lambda Architecture [Marz, Trivadis]

To summarize the concepts of Lambda Architecture:

- All data is sent to both the batch and speed layer
- Master data set is an immutable, append-only set of data
- Batch layer pre-computes query functions from scratch, result is called Batch Views. Batch layer constantly re-computes the batch views.
- Batch views are indexed and stored in a scalable database to get particular values very quickly. Swaps in new batch views when they are available.
- Speed layer compensates for the high latency of updates to the Batch Views.
- Uses fast incremental algorithms and read/write databases to produce real-time views.
- Queries are resolved by getting results from both batch and real-time views.

Event-driven applications require continuous and timely processing of events; the tight latency constraints disallow batching events together for efficiency. Event-driven applications typically involve processing, monitoring, and notification of events [Kambatla]. These applications, involving complex event processing, read as input a stream of events and either (i) independently process individual data items (stream-processing) or (ii) detect complex event patterns (e.g., credit-card fraud detection). While operations on data are independent in stream-processing applications, detecting complex events requires maintaining a state corresponding to past events; i.e., stream-processing is context-free whereas complex event detection is context-sensitive.

3.1 In-memory Computing

Until recently, all analytics—in fact, nearly all database user information access has been through secondary storage. The main reason: the cost of secondary storage has long been a fraction of the cost of primary storage. As demand for Real Time Big Data decision making continues to grow, the cost of primary storage

has been dropping. This has led to broad availability of In-Memory Computing, which retains data in primary storage (RAM) instead of in secondary storage.

The fact that we can store our data in the same address space as our application is the biggest gain. In-Memory Computing relies heavily on that capability, and exposes a new class of complex data processing capabilities that fits well with the distributed nature of the data through real-time map/reduce and stream-based processing as a core element of its architecture. For real time processing of Big Data, Hadoop owes its drawbacks to two main reasons. First, Hadoop was initially designed for batch processing and hence, execution of jobs is not optimized for fast execution. Scheduling, task assignment, code transfer to slaves, and job startup procedures are impossible to be performed in milliseconds. Secondly, the HDFS file system is designed for high throughput data I/O rather than high performance I/O. Data blocks in HDFS are very large and stored on hard disk drives, which with current technology can deliver transfer rates between 100 and 200 megabytes per second.

The first problem can be solved by redesigning job startup and task execution modules. However, even if each machine is equipped with several hard disk modules, the I/O rate would be several hundreds of megabytes per seconds. An elegant solution to this problem is In-Memory Computing. In a nutshell, in-memory computing is based on using a distributed main memory system to store and process big data in real-time.

Main memory delivers higher bandwidth, more than 10 gigabytes per second compared to hard disk's 200 megabytes per second. Access latency is also much better, nanoseconds versus milliseconds for hard disks. Price of RAM is also affordable. Currently, 1 TB of RAM can be bought with less than 20,000$. These performance superiority combined with dropping price of RAM makes in-memory computing a promising alternative to disk-based big data processing. There are few in-memory computing solutions available like: Apache Spark [Spark], GridGain, and XAP. Amongst them, Spark is both open source and free but others are commercial.

In-memory computing does not mean that all the data is kept in memory. Even if a distributed pool of memory is available and the framework uses that for caching of frequently used data, the whole job execution performance can be improved significantly. Efficient caching is especially effective when an iterative job is being executed. Both Spark and GridGain support this caching paradigm. As explained in the previous section, Spark uses a primary abstraction called Resilient Distributed Dataset (RDD) that is a distributed collection of items [Zaharia1]. Spark can be easily integrated with Hadoop and RDDs can be generated from data sources like HDFS and HBase. GridGain also has its own in-memory file system called GridGain File System (GGFS) that is able to work as either a standalone file system or in combination with HDFS, acting as a caching layer. In-memory caching can help in handling huge streaming data.

3.2 Stream Processing

Stream-processing systems operate on continuous data streams e.g., click streams on web pages, user request/query streams, monitoring events, notifications, etc. Stream processing delivers real-time analytic processing on constantly changing data in motion. Analyze first store later! That is the objective of Stream Processing. Stream Processing finds itself heavily used in in equity trading, monitor telecom infrastructure, intelligence in government etc.

Stream-processing systems operate on continuous data streams: e.g., click streams on web pages, user request/query streams, monitoring events, notifications, etc. Figure 2 describes the concept of Stream Processing.

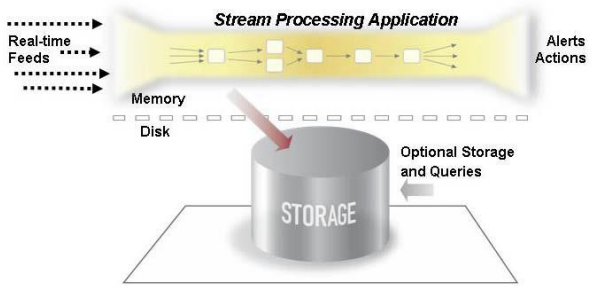

Figure 2. Stream Processing [EMC]

The importance of stream processing systems is increasing with modern applications imposing tighter time constraints on an event's propagation along the pipeline. For instance, Google, Facebook and other advertising companies analyze user-data (e.g., wall post or status change) to display specific ads corresponding to the user's search query. As the the time it takes for ads related to the particular event to be displayed reduces, more data would go through the stream processing pipelines. Stream processing delivers real-time analytic processing on constantly changing data in motion. It enables descriptive and predictive analytics to support real-time decisions [Rea]. Stream computing allows you to capture and analyze all data - all the time, just in time. Data is all around us. From social media feeds to call data records to videos, but can be difficult to leverage. Sometimes there is simply too much data to collect and store before analyzing it. Sometimes by the time we store data, analyze it and respond, it is already too late. From social media feeds to stock market reports to call data records to video streamed by airport cameras and assessed for security purposes, all of these generate continuous streams (sometimes called a Fire Hose) of data. These types of diverse data cannot be easily stored because of their volume, and accurate analysis is no simple matter. And in cases where the data can be stored, by the time it is analyzed, any insights gained will be in the past, often discovered too late to be effective. With stream processing, organizations can react to events as they are happening, enabling them to store less, analyze more, and make better decisions, faster. With stream processing technology, you can continuously analyze massive volumes of your data in memory to take action in real-time. As discussed in an earlier section, IBM introduced InfoSphere Streams [Rea] as a versatile, high-performance and cost effective solution that manage and analyze massive volume, variety and velocity of data that consumers and businesses create every day.

3.3 Complex event processing (CEP)

Complex Event Processing (CEP) processes multiple event streams generated within the enterprise to construct data abstraction and identify meaningful patterns among those streams. CEP along with in-memory data grid technologies, help in pattern detection, matching, analysis and decision making.

Figure 3 describes a typical landscape where Complex Even Processing is used. It would consist of 3 logical units:

- Emitters (Event sources) that's serve as a source for events. Sources can be an application, sensors or even a CEP engine.
- CEP engine is the heart of the systems.
- Targets that are notified under certain event conditions.

After analysis target devices can be alerted or the results can be passed to another CEP engine or the data can be stored.

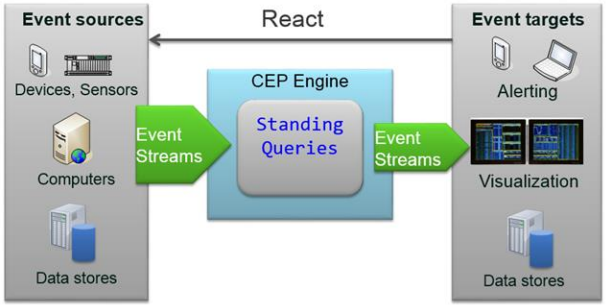

Figure 3. Complex Event Processing [Mayer]

Organizations moving deeper into monitoring operations in real time, results in a growing need to quickly capture and process data expressed as messages or events in a data stream. At the same time, the number of data streams is also increasing because new forms of big data are communicated via streams (e.g., sensor and machine data, plus log and Web data). Furthermore, the greatest analytic insights result from correlating data from multiple streams, as well as from traditional enterprise sources. Examples of applications include financial trading systems, business activity monitoring, utility grid monitoring, e-commerce product recommendations, and facility monitoring and surveillance.

Complex Event Processing addresses two prerequisites for building highly scalable and dynamic systems [Kambatla]:

- CEP decouples providers and receivers of information. The providers do not need any knowledge of the set of relevant receivers and the receivers need not be aware of the corresponding data or event sources.
- CEP-systems not only mediate information in form of events between providers and consumers but also support the detection of relationships among events, for instance, temporal relations that can be specified by definition of correlation rules (sometimes called *Event Patterns*).

Complex event processing has risen in recent years as the preferred method for leveraging streaming data. Technologies for simple event processing have been available for years, but most are designed to monitor only one stream of events at a time. Even if users monitor multiple streams, they end up with multiple, siloed views into real-time business operations. The newer practice of CEP can monitor multiple streams at once while correlating across multiple streams, correlating streaming data with data of other vintages, and continuously analyzing the results. Single-purpose, standalone CEP tools are available from a handful of vendors today. What differentiates CEP from Stream Processing could be that the focus is on unordered streams and on event filtering.

Event Stream Processing, or **ESP**, is a set of technologies designed for 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). In practice, the terms ESP and CEP are often used interchangeably. Stream processing differs from a simple in-memory database in that traditional analytics systems must first load all the data, and then run a query. Streams augments the traditional analytics approach by enabling continuous analysis to be modified over time. Stream computing changes where, when and how much data you can analyze. Store less, analyze more, and make better decisions, faster with stream computing. Stream computing finds most application in Government for security and environmental monitoring and Healthcare etc. What differentiates CEP from Stream Processing could be that the focus is on unordered streams and on event filtering. An event stream is a sequence of events ordered by time, such as a stock market feed. Event stream processing is focused more on high-speed querying of data in streams of events and applying mathematical algorithms to the event data. Some of the first commercial applications were to stock-market feeds in financial systems and algorithmic trading. CEP is focused more on extracting information from clouds of events created in enterprise IT and business systems. CEP includes event data analysis, but places emphasis on patterns of events, and abstracting and simplifying information in the patterns.

4. EXISTING TOOLS

The above technologies have enabled the development of several tools that can be combine with the help of the Lambda Architecture, to build very powerful systems for Big Data Analytics for Real Time Systems. A number of tools like: Hadoop Online, Spark, Streams (IBM), Stinger, Storm (Twitter), Flume, Kafka, Scribe, S4 (Yahoo), Spring XD, IBM Infosphere Streams, HStreaming, Microsoft StreamInsight, Impala, etc., have become popular to analyze volumes of both structured and unstructured data [Liu]. Some of these tools are discussed below.

4.1 Kafka

Kafka was developed as a high performance messaging system. The focus on high performance was from the beginning of its development. It can be scaled simply by adding more computing resources. The architecture consists of Producers that create messages of arbitrary structure, brokers that store the messages and consumers that consume messages (for instance Storm). A 4th component Zookeeper is required because brokers can be arbitrarily added and removed from the Kafka cluster and the zookeper oversees the cluster and can tell producers/consumers about the broker in use. Figure 4 describes Kafka's architecture.

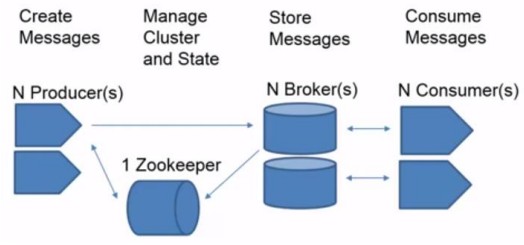

Figure 4. Architecture of Kafka.

Kafka works in combination with Apache Storm, Apache HBase and Apache Spark for real-time analysis and rendering of streaming data. LinkedIn developed Kafka when they moved their data pipeline from a batch-oriented file aggregation mechanism to a real-time publish-subscribe system. Apache Kafka, was developed to power its activity streams, and provides additional reliability guarantee, robust message queueing and distributed publish-subscribe capabilities. This pipeline currently runs in production at LinkedIn and handles more than 10 billion message writes each day with a sustained peak of over 172,000 messages per second. The original use of Kafka is to rebuild a user activity-tracking pipeline as a set of real-time publish/subscribe feeds [Kafka]. Currently, Kafka at LinkedIn supports dozens of subscribing systems and delivers more than 55 billion messages to these consumers each day [Goodhope].

Apache Kafka is a message queue rethought as a distributed commit log. It uses Zookeeper to share and save state between brokers. Each broker maintains a set of partitions: primary and/ or secondary for each topic. A set of Kafka brokers working together will maintain a set of topics. Each topic has its partitions distributed over the participating Kafka brokers and, the replication factor determines, intuitively, the number of times a partition is duplicated for fault tolerance. The architecture of Kafka can be visualized in figure 4.

Many users perform stage-wise processing of data where data is consumed from topics of raw data and then aggregated, enriched, or otherwise transformed into new Kafka topics for further consumption. For example, a processing flow for article recommendation might crawl article content from RSS feeds and publish it to an 'articles' topic; further processing normalizes or de-duplicates this content to a topic of cleaned article content; a final stage might attempt to match this content to users. This creates a graph of real-time data flow out of the individual topics. Apache Storm and Apache Samza are popular frameworks for implementing these kinds of transformations [Kafka]. As documented by LinkedIn [Goodhope], Kafka provides the following:

- Persistent messaging with disk structures that provide constant time performance, even with many TB of stored messages
- High-throughput: even with very modest hardware Kafka can support hundreds of thousands of messages per second
- Explicit support for partitioning messages over Kafka servers and distributing consumption over a cluster of consumer machines while maintaining per-partition ordering semantics
- Support for parallel data load into Hadoop.

LinkedIn activities, such as browse pages, search, and the other user generated actions, are published to a Kafka cluster as topics, one topic per activity type. The topics are available to subscription for a range of use-cases, including real-time processing, real-time monitoring, and loading data into Hadoop or data warehouses. The data pipelines in Kafka are monitored, e.g., monitoring the statistics information of the aggregations from distributed applications for producing centralized feeds of operational data. Therefore, Kafka is well suited for the situations where users need to process real-time data, and analyze them. Apache Storm and Apache Kafka, together have great potential in the real-time streaming market and have so far proven themselves to be very capable systems for performing real-time analytics.

4.2 Storm

is a highly distributed real-time computation system acquired by Twitter. It is fast, scalable, reliable and fault-tolerant. Some Storm concepts:

- Tuple: A tuple is the core unit of data.
- Stream: A sequence of tuples makes a Stream.
- Primitives:
    - Spouts: Pull messages
    - Bolts: Perform core functions of stream computing

Storm needs to pull its input and this is done by a Spout. For example, If Kafka and Storm are being used together then a Spout would pull messages from Kafka. Bolts are individual components that are the core functions of stream computing. The input in Storm flows through a topology and is distributed to run in parallel in the bolts. Figure 5 describes a Storm topology.

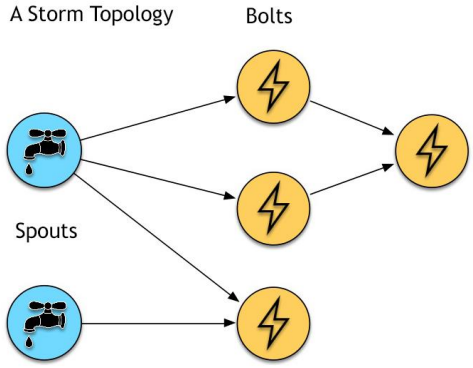

Figure 5. Basic Storm Concepts.

In Storm, a program is defined by two abstractions: Spouts and Bolts. A spout is a source of stream. Spouts can read data from an input queue or even generate data themselves. A bolt process one or more input streams and produces a number of output streams. Most of the process logic is expressed in bolts. Each Storm program is a graph of spouts and bolts which is called a Topology [Toshniwal] [Storm2]. When Twitter user and tweet count began to grow there was a need for a processing scale that didn't exist. To deal with this, Storm was born to provide real-time insight, and process and route their extreme payload of messages. With the ability to process thousands to millions of messages per second with full message guarantees, it provides an amazing engine for real time analytics. Now, this technology has been adopted by a wide range of industries ranging from eCommerce to Financial institutions with great success.

The high level architecture of Storm is shown in Figure 6. Storm consists of several moving components, including a coordinator (ZooKeeper), a state manager (Nimbus), and processing nodes (Supervisor). Storm implements the data flow model in which data flows continuously through a network of transformation entities [Storm]. Storm runs on a distributed cluster. Clients submit topologies to a master node, which is called the Nimbus.

Nimbus is responsible for distributing and coordinating the execution of the topology.

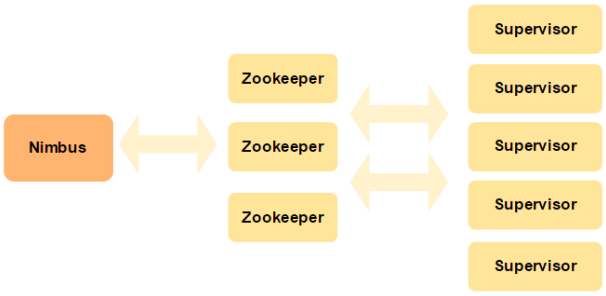

Figure 6. High Level Architecture of Storm.

The abstraction for a data flow is an unbounded sequence of tuples called stream. A tuple is the data structure for representing the standard data types (such as ints, floats, and byte arrays) or user-defined types with some additional serialization code. A stream is used to build data processing topologies between data sources and sinks. Storm's spout abstraction makes it easy to integrate a new queuing system. The key properties of Storm are:

- Fast – Storm is benchmarked as processing one million 100 byte messages per second per node
- Scalable – It is easy to add or remove nodes from the Storm cluster without disrupting existing data flows through Storm topologies (aka. standing queries).
- Fault-tolerant/Resilient – When workers die, Storm will automatically restart them. If a node dies, the worker will be restarted on another node.
- Reliable – Storm guarantees that each unit of data (tuple) will be processed at least once or exactly once. Messages are only replayed when there are failures.
- Easy to operate – standard configurations are suitable for production on day one. Once deployed, Storm is easy to operate.

Today, many enterprises use Storm combined with HBase as their real-time architecture for processing streaming data. In this architecture, Storm is used to integrate streaming data continuously, while HBase is used as the data store for saving data, and for queries [Liu]. Storm is popularly integrated well with Kafka, which is introduced in the next section.

4.3 Spark Streaming

Spark Streaming uses micro-batching to support continuous stream processing. It is an extension of Spark which is a batch-processing system. Apache Spark is an open-source in-memory data analytics cluster computing framework originally developed in the AMPLab at UC Berkeley. In contrast to Hadoop's two-stage disk-based MapReduce paradigm, Spark's in-memory primitives provide performance up to 100 times faster for certain applications. By allowing user programs to load data into a cluster's memory and query it repeatedly, Spark is well suited to machine learning algorithms. For persistence, Spark can use either Hadoop Distributed File System (HDFS) or Cassandra [Zaharia].

Spark Streaming extends the Spark batch processing system to perform stream processing. Many systems require processing the same data in live streaming as well as batch post-processing. Until Spark no single system could do both. Typically we would have to use Hadoop for batch processing at high latency on tera/petabytes of data and the stream processing using Storm at low latency of 100s MB/sec. It is also extremely tedious to maintain these two different stacks of data and deal with the idiosyncrasies of each. Spark streaming runs a streaming computation as a series of very small deterministic batch jobs or small-scale batching. Typical use cases include Website monitoring, Fraud detection, Ad monetization.

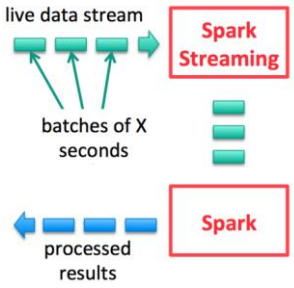

Figure 7. Spark Streaming [Zaharia]

Apache Spark is a general purpose, large-scale processing engine, recently fully inducted as an Apache project and is currently under very active development. The aim of Spark is to make data analytics program run faster by offering a general execution model that optimizes arbitrary operator graphs, and supports in-memory computing. This execution model is called **Resilient Distributed Dataset (RDD)**, which is a distributed memory abstraction of data. However, current-programming models for distributed stream processing are relatively low-level, hence there is lack of consistency of state across the system and fault recovery. Furthermore, the models that provide fault recovery do so in an expensive manner, requiring either hot replication or long recovery times. Zaharia et al., [Zaharia DS] propose a new programming model, discretized streams (D-Streams), that offers a high-level functional programming API, strong consistency, and efficient fault recovery.

4.3.1 Resilient Distributed Datasets

Spark performs in-memory computations on large clusters in a fault-tolerant manner through RDDs [Zaharia1]. Spark can work on the RDDs for multiple iterations, which are required by many machine-learning algorithms. A RDD resides in the main memory, but can be persisted to disk as requested. If a partition of RDD is lost, it can be re-built. Spark also supports shared variable, broadcast variable and accumulator variable. A shared variable is typically used in the situation where a global value is needed, such as lookup tables, and number counter. All transformations, map, join, reduce, etc., in Spark revolve around this type. RDD's can be created in one of three ways: parallelizing (distributing a local dataset); reading a stable, external data source, such as an HDFS file; or transformations on existing RDD's. RDDs let programmers perform in-memory computations on large clusters in a fault-tolerant manner. RDDs are motivated by two types of applications that current computing frameworks handle inefficiently: iterative algorithms and interactive data mining tools.

4.3.2 Discretized Streams

Another type for encapsulating a continuous stream of data: Discretized Streams or DStreams. DStreams are defined as

sequences of RDD's. A DStream is created from an input source, such as Apache Kafka, or from the transformation of another DStream. The key idea behind D-Streams is to treat a streaming computation as a series of deterministic batch computations on small time intervals [Zaharia DS]. D-Streams support a new recovery mechanism that improves efficiency over the traditional replication and upstream backup solutions in streaming databases, parallel recovery of lost state across the cluster.

4.4 Spring XD

Spring XD is a unified, distributed, and extensible system for data ingestion, real time analytics, batch processing, and data export. Spring XD framework supports streams for the ingestion of event driven data from a source to a sink that passes through any number of processors [Spring XD].

To do real-time analytics of big data right, you need data processing support (extraction, transformation, and loading) and an event- and messaging-centric programming model to stitch together otherwise decoupled components in distributed, messaging-centric workflows. This is where Spring XD comes in. Spring XD is a new project in the stream processing space. It builds on the strengths of Spring Batch, Spring Integration, and Spring Data and Spring for Hadoop to meet this new generation of challenges.

We can ingest data from streams or files and in the end we can have both real time views and batch views for further processing. Figure 8 describes Spring XD in use.

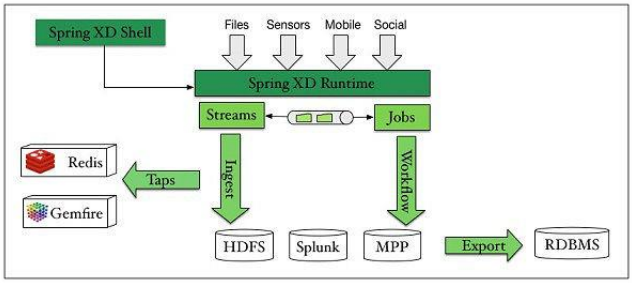

Figure 8. Spring XD Architecture [Spring XD]

As discussed before, Kafka works in combination with Apache Storm, Apache HBase and Apache Spark for real-time analysis and rendering of streaming data. Table 1 is a short comparison between the Spark Streaming, Storm and Spring XD.

Table 1. Comparison of Spark, Storm and Spring XD

	Spark Streaming	**Apache Storm**	**Spring XD**
Definition	A fast and general purpose cluster computing system.	A distributed real-time computation system.	A unified, distributed, and extensible system for data ingestion, real time analytics, batch processing, and data export.
Implemented in	Scala	Clojure	Java
Programming API	Scala, Java, Python	Java API and usable with any programing language.	Java
Development	A full top level Apache project.	Undergoing Apache project.	Spring project by Pivotal.
Processing Model	Batch processing framework that also does micro-batching.	Stream Processing Framework that processes and dispatches messages as soon as they arrive.	Unified platform for stream processing.
Fault Tolerance	Recovery of lost work and restart of workers via the resource manager.	Restart of Workers, Supervisors.	Reassignment of work to container working.
Data processing	Messages are not lost and delivered once. (Small-scale batching)	Keeps track of each and every record.	Unacknowledged messages are retried until the container comes back.
Use Cases	1) Combines batch and stream processing (Lambda Architecture). 2) Machine Learning: Improve performance of iterative algorithms 3) Power Real-time Dashboards	Prevention of: 1) Securities fraud 2) Compliance violations 3) Security breaches 4) Network outage	1) Stream tweets to Hadoop for sentiment analysis. 2) High throughput distributed data ingestion into HDFS from a variety of input sources. 3) Real-time analytics at ingestion time, e.g. gathering metrics and counting values.

4.5 IBM InfoSphere Streams

InfoSphere Streams (or Streams) is an IBM product that supports high performance stream processing [Streams]. It has been used in a variety of sense-and-respond application domains, from environmental monitoring to algorithmic trading. It offers both language and runtime support for improving the performance of sense-and-respond applications in processing data from high rate streams. IBM InfoSphere Streams was started as collaboration between the US Government and IBM. Stream processing extends traditional approaches to information processing, such as transactional or complex event processing (CEP) systems.

Traditional processing involves running analytic queries against historic data. IBM's big data platform Streams processes huge volumes and varieties of data from diverse sources while achieving extremely low latency, enabling decision makers to extract relevant information for timely analysis [Rea]. Streams extends the state of the art in information processing by helping organizations to:

- Respond in real time to events and changing requirements
- Analyze data continuously at rates that are orders of magnitude greater than existing systems
- Adapt rapidly to changing data forms and types
- Deliver real-time cognitive computing functions to learn patterns and predict outcomes
- Manage high availability, heterogeneity and distribution for the stream paradigm
- Provide security and confidentiality for shared information

4.6 Microsoft StreamInsight

StreamInsight is a stream processing system from Microsoft that can be used to develop and deploy temporal complex event processing (CEP) applications. It can be used for clickstream analysis, algorithmic trading, and many other [Microsoft].

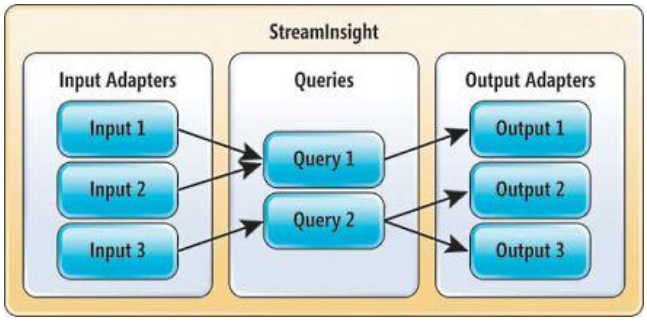

Figure 9. High Level Architecture of StreamInsight [Microsoft].

StreamInsight is used at Microsoft and other companies to perform complex queries on very high volume streaming data and it integrates with the Microsoft .NET framework. StreamInsight provides capability to perform queries using LINQ that join multiple streams and each stream is divided into windows by the CTI (current time increment) event which indicates that no more events will have a start time less than the CTI. It supports multiple types of events including point events, edge events and interval events.

As documented by Microsoft, the need for high-throughput, low-latency processing of event streams is common to the following business scenarios [Microsoft]:

- Manufacturing process monitoring and control
- Clickstream analysis
- Financial services
- Power utilities
- Health care
- IT monitoring
- Logistics
- Telecom

The limitations of StreamInsight are that the data must be windowed by time and that all data in a window must be stored in memory.

4.7 S4

S4 is a general-purpose, distributed, scalable, partially fault-tolerant, pluggable platform that allows programmers to easily develop applications for processing continuous unbounded streams of data. S4 stand for Simple Scalable Streaming System and is a distributed stream processing engine inspired by the MapReduce model. Keyed data events are routed with affinity to *Processing Elements* (PEs), which consume the events and do one or both of the following: (1) emit one or more events which may be consumed by other PEs, (2) publish results [Neumeyer, S4]. S4 provides a runtime distributed platform that handles communication, scheduling and distribution across containers. Distributed containers are called S4 nodes. S4 nodes are deployed on S4 clusters. S4 clusters define named ensembles of S4 nodes. The size of an S4 cluster corresponds to the number of logical partitions (sometimes referred to as tasks). Users develop applications and deploy them on S4 clusters. Applications are built as a graph of:

- Processing elements (PEs)
- Streams that interconnect PEs

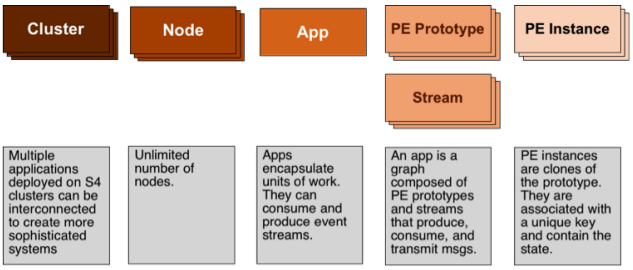

Figure 10. Key concepts of S4 in a hierarchy [S4].

Figure 10 describes the key concepts of S4 in a hierarchical fashion. Figure 10 describes the key concepts of S4 in a hierarchical fashion. PEs communicate asynchronously by sending events on streams. Events are dispatched to nodes according to their key. External streams are a special kind of stream that send events outside of the application, receive events

from external sources and can interconnect and assemble applications into larger systems. Adapters are S4 applications that can convert external streams into streams of S4 events. Since adapters are also S4 applications, they can be scaled easily.

An S4 cluster contains many PEs to process events. Since data is streamed between PEs no on-disk checkpoint is required. Hence S4 supports only partial fault tolerance [Liu]. For example, if a processing node fails, its processes are automatically moved to a standby server, but the states of these processes are lost, and cannot be recovered. This is very similar to Storm except that Storm uses master-slaves architecture instead. Figure 10 describes the concepts of S4.

5. USE CASES

Various use-cases of Big Data Analytics for Real Time Systems in finance, healthcare, government, telecommunication, automotive, retail etc., are discussed below.

5.1 Use Cases in Financial Services

Real Time Analytics of Big Data can help in fighting against cyber threats by gaining real-time insights into security to detect threats as they happen and stop them before damage is done. This real-time actionable insight let financial services quickly identify questionable patterns and stop fraud before it happens. Today banking executives believe in building customer trust, transparency and business capabilities for higher profitability [Groenfeldt]. Banks are launching initiatives with customer dashboards and portals where consumers can get a more customized experience online. The focus is finally shifting to customer experience, therefore leveraging big data and advanced real-time analytics tools to help build better relationships and deliver exactly what customers want. For example, it is interesting to analyze ticks, tweets, satellite imagery, weather trends, and any other type of data to inform trading algorithms in real time [MongoDB].

5.2 Use Cases in Healthcare

Big Data Analytics in Real Time provides a platform for the real-time online analysis of patients' data streams to detect medically significant conditions that precede the onset of medical complications. Patients benefit from the system because earlier detection of signs of the medical conditions may lead to earlier intervention that may potentially lead to improved patient outcomes and reduced length of stays. Some use-cases are described below:

- Healthcare providers can analyze privacy-protected streams of medical device data to detect early signs of disease, identify correlations among multiple patients and measure efficacy of treatments [Rea].
- By combining and analyzing health checkup data and medical history data, the risk of contracting a disease in the future now can be predicted. This information can then be used by healthcare specialists to advise patients and subscribers ways they can improve their health [Fujitsu].

5.3 Use Cases in Transport/Automotive Industry

There are diverse kinds of the traffic data, coming from different kinds of end-users. These end-users include commuters, highway patrols, public service vehicles like fire-engines and ambulances, departments of transportation, urban planners, commercial vehicle operators, etc. These users not only pose large numbers of simultaneous analysis requests, but also require analyses of significantly different natures [Biem]. An important example is the use of GPS for traffic data collection. GPS data and other opportunistic sensors (e.g. smart phones) have great potential to provide the large amounts of data that is needed to support real time management of traffic systems. Some use-cases are described below:

- Streams processes GPS data from buses once every minute, driving a real-time display of all buses as they move through the city [Rea]. Traffic managers can now respond quickly and accurately to relevant insights from real-time analytics drawn from data feeds and reports, enabling them to:
    - Monitor 600 buses across 150 routes daily
    - Analyze 50 bus location updates per second
    - Collect, process and visualize location data for all public transportation vehicles
    - Automatically generate routes and stop locations
- Telematics can provide data-in-motion such as vehicle speed, data relating to the transmission control system, braking, air bags, tire pressure and wiper speed as well as geospatial and current environmental conditions data. Other data contributing to an integrated view of the vehicle includes social media content about customer sentiment and driving experiences. Analysis of this data delivers business benefits by enabling the automotive company to strengthen customer relationships, better predict demand for replacement parts and service, and monetize telematic data [IBM].
- By linking large volumes of traffic information and vehicle information, in terms of location and time, efficient and safe traffic conditions can now be applied through predicting and ascertaining traffic conditions based on actual data [Fujitsu].

5.4 Use Cases in Government

Real time analytics can bring about a revolution in improving public safety by using statistical modeling and real-time social media monitoring to identify and mitigate threats. The era of Big Data is driving considerable changes in how governmental agencies manage and use the varied types of data they acquire and store [Intel]. The legacy Relational Database Management System (RDBMS), Enterprise Data Warehouse (EDW), and Storage Area Network (SAN) infrastructure used by agencies today to create siloed data environments is too rigid to accommodate the demands for massive storage and analyses on a larger and wider variety of data. Some use-cases are described below:

- Identify social program fraud within seconds based on program history, citizen profile, and geospatial data [MongoDB].
- The ability to perform analytics on data-intensive streams lets us identify items or patterns for deeper investigation in Cybersecurity [Rea].

- Using MarkLogic with Hadoop on Intel Architecture, intelligence and law enforcement agencies can extend not only their ability to respond to greater volume, but also better incorporate their unstructured data into the analytical process. Previously, such data has either been too time-consuming to incorporate or simply not possible with traditional technologies [Intel].

5.5 Use Cases in Telecommunication

Big data analytics can be utilized to put real-time intelligence and control back into the network, driving a highly significant increase in capacity and personalize communications to improve customer loyalty based on near real-time analysis of system usage [Banerjee]. Most service providers today are inundated with the explosion of data traffic in their network. However most of this data and usage traffic are not correlated and therefore cannot be used by the service providers. Some use-cases are described below:

- Perform capacity planning for mobile networks as new high-bandwidth services are introduced. Improve customer experience.
- A key Streams [Rea] use case for applications in a wide range of industries: simultaneous processing, filtering and analysis in real time. High availability, automated fault tolerance and recovery—along with real-time dashboard summaries—enhance IT operations.
- Big Data Analytics can help to improve customer profitability analysis, end-to-end visibility for new product rollouts and real-time analysis for better the network customers [Banerjee].

5.6 Use Cases in Retail

Some use-cases of big data analytics in real time for retail and marketing are described below.

- Set up a digital geo-fence around your brick-and-mortar locations to push in-store incentives to shoppers in real time.[MongoDB].
- Evaluate sales performance in real time. Take measures now to achieve sales quotas [Rea].
- Leading retailers monitor the in-store movements of customers, as well as how they interact with products. The rich data feeds are then combined with transaction records and conduct experiments to guide choices about which products to carry, where to place them, and how and when to adjust prices [Brown].
- See a product recurring in abandoned shopping carts. Run a promotion to close more sales of that product [RussomSPLUNK].
- An electric coupon delivery service sends e-mails to customers with recommendations matched to their tastes derived from their location information, membership information, and information on nearby stores [Fujitsu].

5.7 Use Cases in Operational Intelligence

With the help of Operational Intelligence business organisations can gain insight into business opportunities, organizational threats, and performance issues are detected and addressed as soon as possible, thereby enabling reactions that leverage or correct a given situation [RussomSPLUNK]. Some use-cases are described below:

- See a social media sentiment or pattern. Direct it or correct it as it evolves.
- See potentially fraudulent activity while it's in process. Prevent action and proactively mitigate impact.
- See that your utility grid has excess capacity. Sell that capacity while it's available.
- Understand customer behaviour in real time across channels—Web, mobile, and social.

6. DISCUSSION

Major questions arise such as does having big data mean being able to tap valuable information? Does having to invest the resources to overcome issues like privacy, security, and the cost of human talent cancel out the benefits that big data analytics provides?

Recent technological advancements have led to a deluge of data from distinctive domains (e.g., health care and scientific sensors, user-generated data, Internet and financial companies) over the past two decades which has led to an information overload almost everywhere. Big data analytics is trying to take advantage of the excess of information to use it productively. The benefits are many and varied, ranging from higher quality education to cutting-edge medical research, and while further research is needed for things like ensuring people's information is protected from exploitation, there are many exciting discoveries waiting to be uncovered through big data analytics.

A few areas that have been noted as potential Big Data and analytics developments areas over the next few years (Schiller, 2014):

a) An increased level of merging of internal and external data as well as structured and unstructured data.

b) Further applications of social and digital media.

c) More companies starting to create technologies that can handle and process different types of data. This will lead to better targeting of customers, better understanding of customer behavior and risk management processes as well as continuous feedback into the underwriting, pricing and claims processes.

d) There may be an integration and further development around telemetric data.

e) There may be complex statistical methods being introduced with further developed forecasting, machine-learning and data-mining techniques.

f) There may also be a development of self-learning systems to handle the vast amount of data; and

g) The insurance industry may start moving into new products that cover more complex, multiple risks.

h) But certainly according to us: Given these issues, Big Data is a huge opportunity, not a problem.

7. CONCLUSION

This paper introduces Big Data, Big Data Analytics in the context of Real Time Systems. The paper provide a flow of information to create a clear picture from the basics of Big Data to the Use Cases of Big Data in Real Time Systems. An initiative was taken to summarize the major tools and techniques used in Big Data Analytics for Real Time Systems by providing the high level architecture, followed by their advantages and disadvantages. We also described in further detail the Real Time Systems prevalent today and the major Use Cases to impinge on the reader.

The data collection from the various sources stated in the paper began in August 2014 and required effort in understanding the various concepts and presenting them from our perspective. The research papers for Big Data Analytics in Real Time Systems were not many and hence quite a number of technical blogs, whitepapers and articles have been referred to gather the required knowledge.

With the examples described in this paper we wished to demonstrate the different applications in the field of Finance, Automotive, Security and Healthcare etc., where Big Data Analytics in real-time is being widely used today.

Lastly, we would like to thank the department of Learning Technologies Research Group, RWTH for giving us this opportunity and all the authors stated in the references for providing and permitting us to use the necessary information.

8. REFERENCES

[1] Banerjee, Ari. "Addressing" Big Data" Telecom Requirements for Real-Time Analytics." Heavy Reading (2011).

[2] Biem, Alain, et al. "IBM infosphere streams for scalable, real-time, intelligent transportation services." Proceedings of the 2010 ACM SIGMOD International Conference on Management of data. ACM, 2010.

[3] Brown, Brad, Michael Chui, and James Manyika. "Are you ready for the era of 'big data'. "McKinsey Quarterly 4 (2011): 24-35.

[4] Cox, Michael and Ellsworth, David: Application controlled Demand Paging for Out-Of-Core Visualization.

[5] EMC. Stream Processing. Digital image. Community.emc.com. 24 Sept. 2012. Web. URL: https://community.emc.com/servlet/JiveServlet/showImage/38-5549-46648/gspa.png.

[6] Fujitsu, Limited. "Fujitsu Big Data Software Use Cases." (n.d.): 2+. Web. 30 Nov. 2014. http://www.fujitsu.com/global/products/software/middleware/application-infrastructure/interstage/solutions/big-data

[7] Goodhope, Ken, et al. "Building LinkedIn's Real-time Activity Data Pipeline." IEEE Data Eng. Bull. 35.2 (2012): 33-45.

[8] Groenfeldt, Tom. "Banks Betting Big on Big Data and Real-Time Customer Insight." Http://www.sap.com/. SAP, 21 Sept. 2013. Web.

http://public.dhe.ibm.com/common/ssi/ecm/en/iml14293usen/IML14293USEN.PDF

[9] IBM. "IBM Big Data for the Automotive Industry." IBM Big Data for the Automotive Industry (2013): 3. Print.

[10] Intel, and MarkLogic. "Simplifying Data Governance and Accelerating Real-time Big Data Analysis for Government Institutions with MarkLogic Server and Intel." MarkLogic and Intel for Federal, State, and Local Agencies, 2014. Web.

[11] Kafka. URL: http://kafka.apache.org/ as of 2014/12/20.

[12] Kambatla, Karthik, et al. "Trends in big data analytics." Journal of Parallel and Distributed Computing 74.7 (2014): 2561-2573.

[13] Liu, Xiufeng, Nadeem Iftikhar, and Xike Xie. "Survey of real-time processing systems for big data." Proceedings of the 18th International Database Engineering & Applications Symposium. ACM, 2014.

[14] Marron, B.A and P.A.D Maine: Information Retrieval, Washington DC 1967.

[15] Marz, Nathan, and James Warren. Big Data: Principles and best practices of scalable realtime data systems. O'Reilly Media, 2013.

[16] Marz, N., and J. Warren. Lambda Architecture. Digital image. Trivadis, 19 Nov. 2014. Web. URL: http://dbis.cs.unibas.ch/events/copy_of_dbta-workshop-on-semantic-data-processing-07.02.2014-slides/DBTA_Workshop-Streams-2014-12-03-U-Fasoli_Big-Data-Fast-Data.pdf.

[17] Mayer, Manuel. Complex Event Processing Architecture. Digital image. Oct. 2013. Web. URL: http://www.12qw.ch/wp-content/uploads/2013/10/image11.png.

[18] META Group. "3D Data Management: Controlling Data Volume, Velocity, and Variety." February 2001.

[19] Microsoft, Corporation. "Master Large Data Streams with Microsoft StreamInsight." MSDN Magazine. Rob Pierry, June 2011. Web. 20 Dec. 2014. URL: http://msdn.microsoft.com/en-us/magazine/hh205648.aspx.

[20] Microsoft, Corporation. "Microsoft StreamInsight." Microsoft Developer Network. 20 Dec. 2014. URL: http://msdn.microsoft.com/en-us/library/ee391416(v=sql.111).

[21] MongoDB Real-Time Analytics. http://www.mongodb.com/use-cases/real-time-analytics.

[22] Neumeyer, Leonardo, et al. "S4: Distributed stream computing platform." Data Mining Workshops (ICDMW), 2010 IEEE International Conference on. IEEE, 2010.

[23] Raja SP, 2014: The Big Data and Analytics Hub, URL: http://www.ibmbigdatahub.com/blog/big-data-architects-guide-building-real-time-contextual-marketing-systems.

[24] Rea, Roger. "IBM InfoSphere Streams. Redefining Real Time Analytic Processing"." IBM Software, Thought Leadership White Paper (2013): 1-8.

[25] Russom, Philip, 2011: "Big Data Analytics" TDWI best practices report.

[26] Russom, Philip. "Operational Intelligence: Real-Time Business Analytics for Big Data." TDWI Checklist Report:

Operational Intelligence: Real-Time Business Analytics from Big Data (n.d.): 2. TDWI. Web. http://www.splunk.com/web_assets/pdfs/secure/Real-time_Business_Analytics_from_Big_Data.pdf

[27] Russom, Stodder, and Halper. "Real-Time Data, BI, and Analytics." TDWI best practices report, Fourth Quarter (2014).

[28] Schiller, Benjamin Reed 2014: First-Degree Price Discrimination Using Big Data

[29] Spark. URL: spark.apache.org as of 2014/12/18.

[30] Spring XD. URL: http://projects.spring.io/spring-xd/ as of 2015/01/10.

[31] Storm. URL: http://hortonworks.com/hadoop/storm/ as of 2014/12/18.

[32] Storm. URL: storm.apache.org as of 2014/12/16.

[33] Streams. URL: http://www-03.ibm.com/software/products/en/infosphere-streams as of 2014/12/20

[34] S4. URL: http://incubator.apache.org/s4/

[35] Toshniwal, Ankit, et al. "Storm@ twitter." Proceedings of the 2014 ACM SIGMOD international conference on Management of data. ACM, 2014.

[36] Zaharia, Matei, et al. "Discretized streams: an efficient and fault-tolerant model for stream processing on large clusters." Proceedings of the 4th USENIX conference on Hot Topics in Cloud Ccomputing. USENIX Association, 2012.

[37] Zaharia, Matei, et al. "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing." Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation. USENIX Association, 2012.

[38] Zaharia, Matei, et al. "Spark: cluster computing with working sets." Proceedings of the 2nd USENIX conference on Hot topics in cloud computing. 2010.