International Journal of Scientific & Engineering Research, Volume 10, Issue 10, October-2019
ISSN 2229-5518

A Comparative Study on Real Time Data Analysis Frameworks

O. E. Emam
Information Systems Department,
Faculty of Computers and Artificial Intelligence,
Helwan University, Cairo, Egypt.

A. Abdo
Information Systems Department,
Faculty of Computers and Artificial Intelligence,
Helwan University, Cairo, Egypt.

A. M. Abd-Elwahab
Business Information Systems Department,
Faculty of Commerce and Business Administration,
Helwan University, Cairo, Egypt.

Abstract— Today we live in the digital world. With continuous data streaming and increasing digitization the amount of structured and unstructured data being generated from various sources – transactions, social media networks, sensors, machines, mobile phones and other real time sources. The speed and growth of data has affected all fields, whether it is the business sector or the world of science. A large amount of data gives a better output but also working with it can become challenges due to processing frameworks which that one of the essential components of real-time data analysis. So, a real-time platform must meet the needs of data scientists, developers and data center operations teams without requiring extensive custom code or brittle integration of many third-party components. And the data analysis requires scalable, flexible, and high performing tools to provide insights in a timely fashion. This paper presents valuable insights into the understanding of the real-time data analysis frameworks and comparison of popular real-time data processing systems.
Keywords: Big Data, Batch/Stream Processing, Hadoop, Spark, Storm, Samza, Flink.

1 Introduction

Real-time data analytics is the analysis of data as soon as that data becomes available. In other words, users get insights or can draw conclusions immediately or very rapidly after the data enters their system. It's also known as dynamic analysis, real-time analysis, real-time data integration and real-time intelligence [2], [8], [10], [20] and [25].

Real-time analytics allows businesses to react without delay. They can seize opportunities or prevent problems before they happen. By comparison, batch-style analytics may take hours or even days to yield results. Consequently, batch analytical applications often yield only "after the fact" insights (lagging indicators). Business intelligence (BI) Insights from real-time analytics can allow businesses to get ahead of the curve [1], [3] and [12].

The real-time business intelligence is an approach to data analytics that enables business users to get up-to-the-minute data by directly accessing operational systems or feeding business transactions into a real-time data warehouse and business intelligence system. The technologies that can be used to enable real-time BI include data virtualization, data federation, enterprise information integration (EII), enterprise application integration (EAI) and service-oriented architectures (SOA). Complex event processing tools can be used to analyze data streams in real time and either trigger automated actions or alert workers to patterns and trends [3].

Real-time processing of big data mainly focuses on electricity, energy, smart city, intelligent transportation, and intelligent medical fields. During the information processing it needs to be able to make quick decisions, and feedback relevant instructions to the sensing terminal input within a very short time delay [19], [23] and [25].

Analyzing large data sets requires significant compute capacity that can vary in size based on the amount of input data and the type of analysis [16] and [18]. In addition to some systems handle data in batches only mode, while other systems process data in a streaming mode as it flows into the system and others can handle data in a combination of tow-a mixed processing mode [8], [10] and [25].

Processing frameworks and processing engines are responsible for computing over data in a data system. While there is no authoritative definition setting apart "engines" from "frameworks", it is sometimes useful to define the former as the actual component responsible for operating on data and the latter as a set of components designed to do the same. For instance, Apache Hadoop can be considered a processing framework with MapReduce as its default processing engine. Engines and frameworks can often be swapped out or used in tandem. For instance, Apache Spark, another framework, can hook into Hadoop to replace MapReduce. This interoperability between components is one reason that big data systems have great flexibility [6], [10], [14] and [15].

Therefore, the research of real-time data analysis has a great application prospect and research value. Because of the real-time analytics of data processing framework is one of the essential components of a big data analytics.

This paper is organized as follows: Section 2 in this paper, examines the related work of real-time data analytics. In section 3, the real-time data analytics systems are discussed. In addition, other real-time data analysis tools are summarized in section 4. In section 5, comparison of popular real-time data processing systems. Finally, conclusion and directions for future work are reported in section 6.

2 Related Work

Recent years, many experts and scholars have made a lot of research on real-time analytics [3], [4], [5] and [10].

Especially on real-time streaming data processing [9], [11], [16], [17] and [22], in addition to many scholars' research on real-time big data processing is in full swing [2], [7], [8], [20], and [25].

Many researchers have made studies on a framework for real-time data processing [2], [5], [10], [13], [14], [17], [21] and [25].

International Journal of Scientific & Engineering Research, Volume 10, Issue 10, October-2019
ISSN 2229-5518

In [9], Gürcan and Berigel, provided a valuable insight on real-time processing of big data streams, with its lifecycle, tools and tasks and challenges. This paper initially revealed the lifecycle of real-time big data processing, consisting of four phases, that are data ingestion, data processing, analytical data store, and analysis and reporting. Secondly, it described tools and tasks of real-time big data processing. These tools are: Flume, Kafka, Nifi, Storm, Spark Streaming, S4, Flink, Samza, Hbase, Hive, Cassandra, Splunk, and Sap Hana. Finally, challenges of real-time big data processing were identified and categorized.

In [13], Inoubli, et al., proposed streaming frameworks for Big Data applications to store, analyze and process the continuously captured data. Also discussed the challenges of Big Data and presented an experimental evaluation and a comparative study of the most popular streaming platforms.

In [14], Inoubli, et al., surveyed popular frameworks for large-scale data processing. This paper presented an overview of the Big Data frameworks Hadoop, Spark, Storm and Flink. Also, presented a categorization of these frameworks according to some main features such as the used programming model, the type of data sources, the supported programming languages and whether the framework allows iterative processing or not. Finally, conducted an extensive comparative study of the above presented frameworks on a cluster of machines and highlighted best practices while using the studied Big Data frameworks.

In [17], Khan, et al., proposed a framework for the dynamic visualization of real time streaming big data, resilient to both its volume and rate of change. Some of the different directions explored include: (a) the efficient processing and consumption of streaming data; (b) the automated detection of relevant changes in the data stream, highlighting entities that merit a detailed analysis; (c) the choice of the best idioms to visualize big data, possibly leading to the development of new visualization idioms; (d) real-time visualization changes.

In [21], Yadranjiaghdam et al., proposed a framework for real-time analysis of Twitter data. This framework is designed to collect, filter, and analyze streams of data and gives us an insight to what is popular during a specific time and condition. The framework consists of three main steps; data ingestion, stream processing, and data visualization components with the Apache Kafka messaging system that is used to perform data ingestion task. Finally, conducted a case study on tweets about the earthquake in Japan and the reactions of people around the world with analysis on the time and origin of the tweets.

In [16], Jayanthi and Sumathi, focused on the challenges that real-time stream processing solution addressed using machine learning. Also, this paper analyzed the traditional analytic tools to bridge the gap between data being generated and data that can be analyzed effectively.

In [2], Anjos, et al., presented a framework consisting of composable data-analysis services that can be combined to address needs of specific applications. Also, this paper focused on applications for small and medium-sized organizations, the framework offered a flexible and lightweight approach that allows these organizations to take advantage of Big Data analysis in the cloud infrastructures.

In [25], Zheng, et al., built a kind of real-time big data processing (RTDP) architecture based on the cloud computing technology and then proposed the four layers of the architecture, and hierarchical computing model. Also, proposed a multi-level storage model and the LMA-based application deployment method to meet the real-time and heterogeneity requirements of RTDP system. Then used DSMS, CEP, batch-based MapReduce and other processing mode and FPGA, GPU, CPU, ASIC technologies differently to processing the data at the terminal of data collection. Finally, structured the data and upload to the cloud server and MapReduce the data combined with the powerful computing capabilities cloud architecture.

3 Real-Time Data Analytics Systems

In this section, the real-time data analysis systems presented in a table of comparison as shown in Table 1. As shown in Table 1, the comparison analysis of the frameworks, tools and tasks, Issues, challenges and solutions, advantages and limitations involve three domains, which are batch, stream and Hybrid processing systems. In this respect, a total of 21 related works have been analyzed, in which eight related works were identified within the domain of batch processing systems and another seven in the stream processing systems domain. However, six works were found in the Hybrid processing systems domain. On the other side, this finding indicates eleven related works interested in the frameworks and tasks for real-time analytics and seven related works in challenges and solutions. finally, three related works in advantages and limitations.

TABLE 1
Real-Time Data Anaytics Systems

Aspects	Batch-Only Processing Systems	Stream-Only Processing Systems	Hybrid Processing Systems	Total
Frameworks, Tools, and Tasks	In [8], *Dugane & Raut*, 2014.	In [13], *Inoubli, et al.*, 2018.	In [14], *Inoubli, et al.*, 2018.	11
	In [4], *Bifet*, 2013.	In [17], *Khan, et al.*, 2017.		
	In [24], *Zhang, et al.*, 2010.	In [21], *Yadranjiaghdam, et al.*, 2017.	In [2], *Khan, et al.*, 2015.	
	In [3], *Azvine*, 2006.	In [16], *Jayanthi & Sumathi*, 2016.	In [20], *Babak, et al.*, 2017.	

International Journal of Scientific & Engineering Research, Volume 10, Issue 10, October-2019
ISSN 2229-5518

Issues, Challenges, and Solutions	In [7], *Brock & Khan*, 2017. In [12], *Gandhi & Sreedhar*, 2015. In [15], *Jaseena, et al.*, 2014.	In [11], *Hiraman, et al.*, 2018. In [9], *Gürcan & Berigel*, 2018.	In [25], *Zheng, et al.*, 2015. In [23], *Yuan, et al.*, 2012.	**7**
Advantages and Limitations	In [1], *Anjos, et al.*, 2018.	In [22], *Yang, et al.*, 2013.	In [10], *Gurusamy, et al.*, 2017.	**3**
Total	**8**	**7**	**6**	**21**

3.1 Batch Processing Systems

Batch processing is the processing of vast amounts of data sets that is collected over a period of time and return the result at a later time when the computation is completed. The dataset easily consists of millions of records for a day and can be stored in a variety of ways (file, record, etc.). An example of a batch processing job is all of the transactions a financial firm might submit over the course of a week. It can also be used in payroll processes, line item invoices, and supply chain and fulfillment.

Batch data processing is an extremely efficient way to process large volume of data all at once. It also helps to reduce the operational costs that businesses might spend on labor as it doesn't require specialized data entry clerks to support its functioning. It can be used offline and gives managers complete control as to when to start the processing, whether it be overnight or at the end of a week or pay period.

3.1.1 Apache Hadoop

Apache Hadoop is a collection of open-source software for reliable, scalable, distributed computing that facilitate using a network of many computers to solve problems involving massive amounts of data and computation.

Apache Hadoop is a processing framework that exclusively provides batch processing. It can be used for the distributed storage and processing of large data sets across clusters of computers using simple programming models. Batch processing works well in situations where you don't need real-time analytics results, and when it is more important to process large volumes of data to get more detailed insights than it is to get fast analytics results. The base Apache Hadoop framework is composed of the following modules:

1. **Hadoop Common** – contains libraries and utilities needed by other Hadoop modules.
2. **Hadoop Distributed File System** (**HDFS**) – a distributed file system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster nodes.
3. **Hadoop Yet Another Resource Negotiator (YARN)** – a platform responsible for managing computing resources in clusters and using them for scheduling users' applications.
4. **Hadoop MapReduce** – an implementation of the MapReduce programming model for large-scale data processing. MapReduce is Hadoop's native batch processing engine. It's the best framework for processing data in batches.

Batch Processing Model:

The processing functionality of Hadoop comes from the MapReduce engine. MapReduce's processing technique follows the *map*, *shuffle* and *reduce* algorithm using key-value pairs.

The basic procedure involves:

- Reading the dataset from the HDFS file system.
- Dividing the dataset into chunks and distributed among the available nodes.
- Applying the computation on each node to the subset of data (the intermediate results are written back to HDFS).
- Redistributing the intermediate results to group by key.
- "Reducing" the value of each key by summarizing and combining the results calculated by the individual nodes.
- Write the calculated final results back to HDFS.

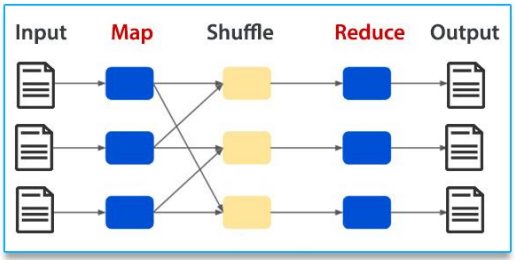

Figure 1. The Framework of the MapReduce Model [24]

3.2 Stream Processing Systems

Stream processing is computing over data and produce the results strictly within certain time constrains as data enter the system and quickly detect conditions within a small time period from the time of receiving the data. The detection time period may vary from a few milliseconds to minutes. This requires a different processing model than the batch processing paradigm. Stream processing is also known as real-time analytics, streaming analytics, complex event processing, real-time streaming analytics, and event processing. Stream processing is a good fit for data where you must respond to changes or spikes and where you're interested in trends over time.

International Journal of Scientific & Engineering Research, Volume 10, Issue 10, October-2019
ISSN 2229-5518

3.2.1 Apache Storm

Apache storm is a free and open-source distributed real-time computational system for processing data streams. Storm focuses on extremely low latency and it can handle very large quantities of data with and deliver results with less latency than other solutions so it's considered one of the best options for workloads that require near real-time processing.

Storm has many use cases: real-time analytics, online machine learning, continuous computation, distributed RPC, ETL, and more. Storm is fast: a benchmark clocked it at over a million tuples processed per second per node. It is scalable, fault-tolerant, guarantees your data will be processed, and is easy to set up and operate.

Storm Processing Model:

Storm integrates with the queueing and database technologies you already use. Stream processing works by orchestrating DAGs (Directed Acyclic Graphs) in a framework it calls topologies. A Storm topology consumes streams of data and processes those streams in arbitrarily complex ways, repartitioning the streams between each stage of the computation however needed. These topologies describe the various transformations or steps that will be taken on each incoming piece of data as it enters the system.

The topologies are composed of:

- **Streams**: Conventional data streams. This is unbounded data that is continuously arriving at the system.
- **Spouts**: Sources of data streams at the edge of the topology. These can be APIs, queues, etc. that produce data to be operated on.
- **Bolts**: Bolts represent a processing step that consumes streams, applies an operation to them, and outputs the result as a stream. Bolts are connected to each of the spouts and then connect to each other to arrange all of the necessary processing. At the end of the topology, final bolt output may be used as an input for a connected system.

3.2.2 Apache Samza

Apache Samza is an open-source distributed near-realtime, asynchronous computational framework for stream processing. It's based on Apache Kafka and YARN. It provides a simple callback-based API that's similar to MapReduce, and it includes snapshot management and fault tolerance in a durable and scalable way.

Samza is designed specifically to take advantage of Kafka's unique architecture and guarantees. It uses Kafka to provide fault tolerance, buffering, and state storage. Samza uses YARN for resource negotiation. This means that by default, a Hadoop cluster is required (at least HDFS and YARN), but it also means that Samza can rely on the rich features built into YARN.

Samza allows you to build stateful applications that process data in real-time from multiple sources including Apache Kafka. Battle-tested at scale, it supports flexible deployment options to run on YARN or as a standalone library.

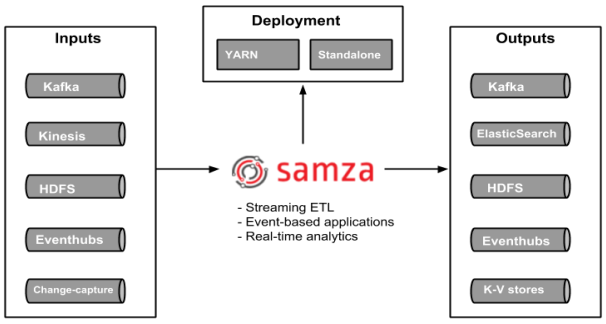

Figure 2. The Framework of the Apache Samza Model [9]

Samza Processing Model:

Samza relies on Kafka's semantics to define the way that streams are handled. Kafka uses the following concepts when dealing with data:

- **Topics**: Each stream of data entering a Kafka system is called a topic. A topic is basically a stream of related information that consumers can subscribe to.
- **Partitions**: In order to distribute a topic among nodes, Kafka divides the incoming messages into partitions. The partition divisions are based on a key such that each message with the same key is guaranteed to be sent to the same partition. Partitions have guaranteed to order.
- **Brokers**: The individual nodes that make up a Kafka cluster are called brokers.
- **Producer**: Any component writing to a Kafka topic is called a producer. The producer provides the key that is used to partition a topic.
- **Consumers**: Consumers are any component that reads from a Kafka topic. Consumers are responsible for maintaining information about their own offset, so that they are aware of which records have been processed if a failure occurs.

3.3 Hybrid Processing Systems (Batch and Stream Processing)

There are some processing systems can handle both batch and stream frameworks. These frameworks simplify diverse processing requirements by allowing the same or related components and APIs to be used for both types of data.

3.3.1 Apache Spark

Apache Spark is an open-source distributed general-purpose cluster-computing framework that is a unified analytics engine for large-scale data processing and the next generation batch processing framework with stream processing capabilities.

International Journal of Scientific & Engineering Research, Volume 10, Issue 10, October-2019
ISSN 2229-5518

Apache Spark achieves high performance for both batch processing workloads by offering full in-memory computation and processing optimization and streaming data, using a state-of-the-art DAG scheduler, a query optimizer, and a physical execution engine.

Spark can run as a standalone or on top of Hadoop YARN, where it can read data directly from HDFS. In addition to its in-memory data processing engine that can do ETL, analytics, machine learning and graph processing on large volumes of data at rest (batch processing) or in motion (streaming processing) with rich concise high-level APIs for the programming languages: Scala, Python, Java, R, and SQL.

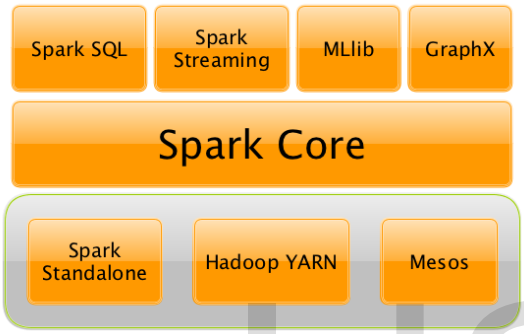

Figure 3. The Spark Platform [14]

3.3.2 Apache Flink

Apache Flink is an open-source stream-processing framework and distributed processing engine for stateful computations over unbounded and bounded data streams. Flink has been designed to run in all common cluster environments, perform computations at in-memory speed and at any scale.

The core of Apache Flink is a distributed streaming dataflow engine written in Java, Scala, Python and SQL and are automatically compiled and optimized into dataflow programs that are executed in a cluster or cloud environment.

Flink provides a high-throughput, low-latency streaming engine as well as support for event-time processing and state management. Flink applications are fault-tolerant in the event of machine failure and support exactly-once semantics.

Apache Flink is an excellent choice to develop and run many different types of applications due to its extensive features set. Flink's features include support for stream and batch processing, sophisticated state management, event-time processing semantics, and exactly-once consistency guarantees for state. Moreover, Flink can be deployed on various resource providers such as YARN, Apache Mesos, and Kubernetes but also as stand-alone cluster on bare-metal hardware. Configured for high availability, Flink does not have a single point of failure. Flink has been proven to scale to thousands of cores and terabytes of application state, delivers high throughput and low latency, and powers some of the world's most demanding stream processing applications.

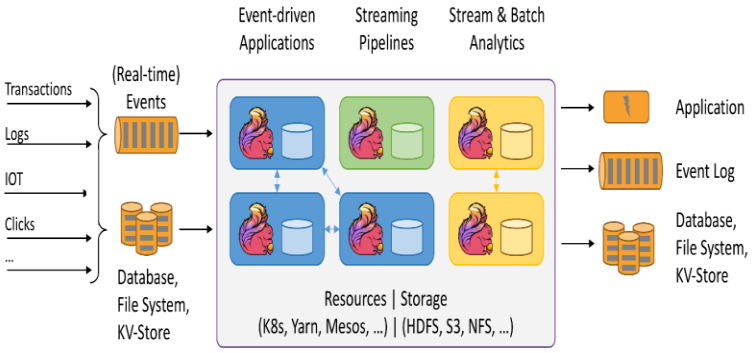

Figure 4. The Flink Platform [2]

4 Other Real-Time Data Analysis Tools

4.1. Apache Flume is a distributed, reliable, and available software for efficiently collecting log data present in log files from web servers, aggregating it in HDFS for analysis, and moving large amounts of log data into HDFS. It has a simple and flexible architecture based on streaming data flows. It is robust and fault tolerant with tunable reliability mechanisms and many failover and recovery mechanisms. It uses a simple extensible data model that allows for online analytic application.

4.2. Apache Kafka is an open-source distributed stream-processing software platform that is used publish and subscribe to streams of records. It is horizontally scalable, fault-tolerant, wicked fast, and runs in production in thousands of companies. Kafka is designed to allow your apps to process records as they occur.

4.3. Apache S4 is a general-purpose, near real-time, distributed, decentralized, scalable, event-driven, modular platform that allows programmers to easily implement applications for processing continuous unbounded streams of data. S4 has a decentralized and symmetric architecture which all the nodes in a cluster are identical, different to the classic master-nodes architecture. S4 employs ZooKeeper as the communication layer to coordinate the nodes within the cluster.

4.4. Apache NiFi is an open-source software project from the Apache Software Foundation designed to automate the flow of data between software systems. It is based on the "Niagara Files" software previously developed by the NSA. Apache NiFi supports powerful and scalable directed graphs of data routing, transformation, and system mediation logic.

4.5. Apache Hive is a data warehouse software project built on top of Apache Hadoop for providing data query and analysis. It's facilitates reading, writing, and managing large datasets residing in distributed storage using SQL. Structure can be projected onto data already in storage. A command line tool and JDBC driver are provided to connect users to Hive.

International Journal of Scientific & Engineering Research, Volume 10, Issue 10, October-2019
ISSN 2229-5518

4.6. **Apache Sqoop** is a command-line interface application designed for efficiently transferring bulk data between Apache Hadoop and structured datastores such as relational databases.

4.7. **Apache HBase** is the Hadoop database, a distributed, scalable, big data store. It is used when you need random, realtime read/write access to your Big Data. This project's goal is the hosting of very large tables -- billions of rows X millions of columns -- atop clusters of commodity hardware. HBase is an open-source non-relational distributed database modeled after Google's Bigtable and written in Java.

4.8. **HStreaming** is an analytics platform built on top of Hadoop and MapReduce. The architecture of HStreaming consists of two components: data acquisition and data analytics. The data acquisition component is able to collect data in near real-time, and has ETL capabilities, while the analytics component allows to analyze unstructured and structured data on HDFS in a real-time fashion.

4.9. **Apache Impala** is a modern, open source massively parallel processing, distributed SQL query engine for data stored in a computer cluster running Apache Hadoop.

5 Comparison of Real-Time Data Processing Systems

The frameworks for real-time data analysis presented above are compared according to several features (see Table 2) including: data format, types of data sources, Architecture, specified used for RTDP, processing model, programming model, supported languages, cluster manager, integration/query, Comments, latency, messaging capacities, iterative computation, interactive mode, auto-parallelization, data partitioning and data transport.

TABLE 2
Comparison of Popular Real-Time Data Processing Systems

Aspects	Hadoop	Storm	Samza	Spark	Flink
Data Format	Key-value	Key-value, Tuples	Events, Messages	Key-value, RDD	Key-value, Data stream
Data Sources	HDFS, YARN and MapReduce	HDFS, HBase and Kafka	Kafka	HDFS, DBMS and Kafka	Kafka, Kinesis, message queus, socket streams and files
Architecture	Master/Slaves	Peer	Peer	Master/Slaves	Publish/Subscribe
Specified Used	Data ingestion, Single-event	Stream & Complex event processing	Event stream processing	Event stream processing	Stream & Complex event processing
Processing Model	Batch	Stream	Stream	Batch and Stream	Batch and Stream
Programming Model	Map and Reduce	Topology	Map and Reduce	Transformation and Action	Transformation, Action Functions
Supported languages	Java, Python and R	Java, Scala, Clojure, Python, Ruby and C#	Java, Scala, Python and Ruby	Java, Scala, Python and R	Java, Scala, Python, Ruby, R and Clojure
Primarily Written in	Java	Clojure	Java, Scala	Scala	Java, Scala
Cluster Manager	YARN	Zookeeper or YARN	YARN	Standalone, YARN and Apache Mesos	Zookeeper
Integration/Query (Streaming query)	Integration	Integration	Integration and Samza-SQL API	Integration and Spark-SQL	Integration
Comments	Store large data in HDFS	Suitable for real time App.	Based on Hadoop and Kafka	Gives several APIs to develop interactive App.	Extension of MapReduce with Graph methods
Latency	More Seconds	Sub-Second	Few Seconds	Sub-Second	Sub-Second
Messaging Capacities	At least once	At least once	Exactly once	Exactly once	Exactly once
Iterative Computation	Yes (By running multiple Map-Reduce jobs)	Yes	Yes	Yes	Yes
Interactive Mode	No	No	No	Yes	No
Auto-Parallelization	On demand	Pipelined Processing	On demand	On demand	Pipelined Processing
Data Partitioning	Yes	No	Yes	Yes	No
Data Transport	RPC	RPC	Kafka	RPC	RPC

International Journal of Scientific & Engineering Research, Volume 10, Issue 10, October-2019
ISSN 2229-5518

6 CONCLUSION

Real-time data analysis for computing over data and reduce the results is undoubtedly a huge challenge. There are many solutions to address big data analytic requirements. Most of this solution use multiple systems to build a complete solution. Most of the frameworks for real-time data analysis use multiple systems to build a complete solution. this solution can help to meet the strict business requirements in the most cost-optimized, performant, and resilient way possible. The result is a flexible, big data architecture that is able to scale along with your business on the global infrastructure. There are many options for real-time data processing within a big data system.

First, for batch-only workloads that are processing large sets of data and not time-sensitive, Hadoop is a good choice that is great for MapReduce data analysis on huge amounts of data and likely less expensive to implement than some other solutions.

Second, for stream-only workloads, Storm is probably the best solution available for real-time processing. It's able to handle data with extremely low latency processing and has wide language support. Samza integrates tightly with YARN and Kafka in order to provide flexibility, easy multi-team usage, and straightforward replication and state management.

third, for hybrid processing systems, Spark is a good stream processing solution for workloads that value throughput over latency. It's provided high-speed batch processing, micro-batch processing, wide support, integrated libraries and tooling, and flexible integrations. Flink is currently a unique option in the processing framework world. It's provided true stream processing with batch processing support, low latency processing, high throughput and real entry-by-entry processing but is still in the early days of adoption.

Finally, the best fit for your situation will depend heavily upon the state of the data to process, how time-bound your requirements are, and what kind of results you are interested in.

REFERENCES

[1] J. Anjos, K. Matteussi, P. Souza, C. Geyer, A. Silva Veith, G. Fedak, J. Barbosa, "*Enabling Strategies for Big Data Analytics in Hybrid Infrastructures*", International Conference on High Performance Computing & Simulation (HPCS), Orléans, France, (2018), 869-876.

[2] J. Anjos, M. Assuncao, J. Bez, C. Geyer, E. P. Freitas, A. Carissimi, J. Costa, G. Fedak, F. Freitag, V. Markl, P. Fergus, R. Pereira, "*An Application Framework for Real Time Big Data Analysis on Heterogeneous cloud Environment*", IEEE International Conference on Computer and Information Technology, Liverpool, United Kingdom (2015), 199-206.

[3] B. Azvine, Z. Cui, D. Nauck, B. Majeed, "*Real Time Business Intelligence for the Adaptive Enterprise*", International Conference on E-Commerce Technology, San Francisco, CA, USA, (2006), 1-12.

[4] A. Bifet, "*Mining Big Data in Real Time*", International Journal of Informatica, 37 (2013), 15-20.

[5] A. Block, B. Brandenburg, J. H. Anderson, S. Quint, "*An Adaptive Framework for Multiprocessor Real-Time Systems*", Euromicro Conference on Real-Time Systems, Prague, Czech Republic, (2008), 1-12.

[6] K. Boukhelfa, F. Belala, "*Towards a Formalization of Real-Time Patterns-Based Designs*", International Conference on IFIP Advances in Information and Communication Technology, (2015), 625-636.

[7] V. Brock, H. Khan, "*Big data analytics: does organizational factor matters impact technology acceptance?*", Journal of Big Data, (2017), 4-21.

[8] R. Dugane, A. Raut, "*A Survey on Big Data in Real Time*", International Journal on Recent and Innovation Trend in Computing and Communication, 2 (4) (2014), 794-797.

[9] F. Gürcan, M. Berigel, "Real-Time Processing of Big Data Streams: Lifecycle, Tools, Tasks, and Challenges", IEEE 2nd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), (2018), 1-6.

[10] V. Gurusamy, S. Kannan, K. Nandhini, "*The Real Time Big Data Processing Framework: Advantages and Limitations*", International Journal of Computer Sciences and Engineering, 5 (12) (2017), 305-311.

[11] B. Hiraman, C. Viresh M., K. Abhijeet C., "*A Study of Apache Kafka in Big Data Stream Processing*", International Conference on Information, Communication, Engineering and Technology, India, (2018), 1-3.

[12] K. Indra Gandhi, Sri. C. Sreedhar, "*Survey on Big Data: Management and Challenges*", International Journal of Computer Trends and Technology (IJCTT), 20 (1) (2015), 33-36.

[13] W. Inoubli, S. Aridhi, H. Mezni, M. Maddouri, E. Nguifo, "*A Comparative Study on Streaming Frameworks for Big Data*", 44th International Conference on Very Large Data Bases: Workshop LADaS - Latin American Data Science, (2018), Rio de Janeiro, Brazil.1-8.

[14] W. Inoubli, S. Aridhi, H. Mezni, M. Maddauri, E. M. Nguifo, "*An Experimental Survey on Big Data Frameworks*", International Journal of Future Generation Computer Systems-Elsevier, (2018), 1-21.

[15] K. Jaseena, J. David, "*Issues, Challenges, and Solutions: Big Data Mining*", International Journal of Computer Science & Information Technology, (2014), 131-140.

[16] D. Jayanthi, G. Sumathi, "*A Framework for Real-time Streaming Analytics using Machine Learning Approach*", International Journal of Advanced Computer Technology, (2016), 85-91.

[17] A. Khan, D. Gonçalves, D. C. Leão, "*Towards an Adaptive Framework for Real-Time Visualization of Streaming Big Data*", Eurographics International Conference on Visualization (EuroVis), Posters Track, (2017), 1-3.

[18] V. Ta, C. Liu, G. Wandile, "*Big Data Stream Computing in Healthcare Real-Time Analytics*", IEEE International Conference on Cloud Computing and Big Data Analysis, Chengdu, China, (2016).

[19] R. Wagensveld, U. Margull, "*Experiences with HPX on embedded real-time systems*", International Conference on Applied Electronics, Pilsen, Czech Republic, (2017), 1-6.

[20] B. Yadranjiaghdam, N. Pool, N. Tabrizi, "*A Survey on Real-Time Big Data Analytics: Applications and Tools*", IEEE International Conference on Computational Science and Computational Intelligence, Greenville, NC, USA (2016), 404-409.

[21] B. Yadranjiaghdam, S. Yasrobi, N. Tabrizi, "*Developing a Real-Time Data Analytics Framework for Twitter Streaming Data*", IEEE 6th International Congress on Big Data, (2017), 329- 336.

[22] W. Yang, X. Liu, L. Zhang, and L. T. Yang, "*Big Data real-time processing based on Storm*", IEEE 12th International Conference on Trust, Security and Privacy in Computing and Communications, (2013), 1784–1787.

[23] Q. Yuan, Z. Feng, F. Jun, M. Qiang, "*Real-time processing for high speed data stream over large scale data*", Chinese Journal of Computers, 35 (3) (2012), 477-490.

[24] F. Zhang, J. Cao, X. Song, H. Cai, C. Wu, "*An Adaptive MapReduce Framework for Real Time Applications*", IEEE 9th International Conference on Grid and Cloud Computing, Nanjing, China, (2010), 1-6.

[25] Z. Zheng, P. Wang, J. Liu, S. Sun, "*Real-Time Big Data Processing Framework: Challenges and Solutions*", International Journal of Applied Mathematics & Information Sciences, 9 (6) (2015), 3169-3190.