International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 04 | Apr 2020 www.irjet.net p-ISSN: 2395-0072

A Review on Data Streaming and Frameworks used in Data Stream Processing

Yogeshwari, Suma B

[1]*Department of Computer Science and Engineering, R.V College of Engineering, Bengaluru, Karnataka, India*
[2]*Professor, Department of Computer Science and Engineering, R.V College of Engineering, Bengaluru, Karnataka, India*

Abstract – *With the emergence of cloud computing, big data analytics, and Internet of Things (IoT), there is an increasing analysis in the data streaming processes for making real-time data-driven decisions. Data streaming is a process in which large amount of data is transferred continuously. It allows real-time data to be analyzed. Some of the examples of data streaming are IoT sensors, real-time advertising, click-stream data from websites and apps, server and security logs, wireless communications, etc. In this paper, we present an overview of the state-of-the-art technologies in the area of data stream processing. Also, a brief evolution of data streaming process will be discussed along with an overview of two important frameworks, namely Apache Storm and Apache Flink.*

Key Words: Data Stream Processing, Data Streaming Apache Storm, Apache Flink, Data stream processing frameworks

1.INTRODUCTION

In this age of Information Technology, with such advancement, huge quantity of data is being generated on a daily basis, for example, search engines, user data in social media, emails, computer logs, messages, wireless communications, etc. With this vast amount of data being generated in real-time, there is a need to process this high-velocity and real-time data on a continuous basis, contributing to variety of technical challenges.

1.1 Data Streaming and its Evolution

Data streaming is the processing of huge amount of data being generated at high-velocity from various sources in real-time. This paradigm is applicable to high-velocity data being generated from various sources including Internet of Things, click-stream, business intelligence, market data, social media, mobile phones, etc.

With the invention of computers, the need for processing of information and data was created. In the early days, computer scientists used to write programs to process data, which used to be stored on punch cards. Later, Assembly languages and programming languages like Fortran, C, C++ and Java were brought into existence. These languages were used by software engineers to programs for the tasks of data processing. Consequently, databases like SQL, IBM's database were invented, which improved the recognition of data processing by broader audiences. This helped more people to reach data processing, and therefore no longer need to depend on programmers to for generating specific programs and analyze the data. SQL had also extended the amount and form of applications related to data processing such as enterprise applications, year-on-year development estimates, average basket size, etc.

Batch processing was being used for data processing, in which the data used to be stored in a storage system and computations used to be scheduled on these systems to process the data. These computations were not run continuously, as a result of which, the computations would have to be re-run on a continuous basis to get up-to-date results. The age of Big Data emerged with the paper on MapReduce, published by Google. MapReduce is a programming tool used for processing and creating large data sets [3]. This programming model was based on two primitives – Map and Reduce. Users need to specify a map function, which generates a set of intermediate key/value pairs by processing a key/value pair. They also need to specify reduce function which merges all the values which are associated with an intermediate key. This functional style of programming allowed computations to be parallelized and also allowed the executions of these programs on large clusters of commodity machines. Later, Apache Hadoop came into picture. It is an open-source framework for storing and processing large data sets in distributed computing environment. After Apache Hadoop, the field of data processing evidenced the emergence of Apache Spark, which took parallelism in batch processing to next level. Then Apache Storm came into picture, which introduced stream processing. It enabled programs to be written such that they can be run continuously. Unlike batch processing, in stream processing the programs can be run on the data continuously and outcomes produced in real-time, while generating data. Later stream processing advanced with the introduction of Apache Kafka, which brought storage mechanism in message streams. It played the role of buffer in between the sources of data and processing systems. As the initial adopters didn't

think of streaming process as reliable enough, they used both batch processing and stream processing simultaneously. Lambda architecture was such a system, which was a combination of both the processing systems. Recently, Apache Flink was introduced, with the introduction of which, developers and data/ analytics leaders started believing that stream processing can be trusted with mission critical applications, processing of complex events, and continuous running. [6]

2. OVERVIEW OF DATA STREAMING FRAMEWORKS

2.1 Apache Storm

Storm was developed by Nathan Marz and a team at BackType, which was later acquired by Twitter, who made the project open sourced [7]. Later it was adopted by Apache and it became the benchmark for distributed real-time processing systems, enabling to process a large amount of data.

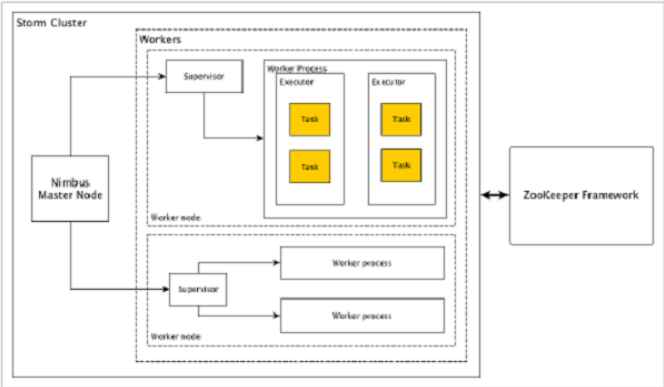

Fig-1: Cluster design of Apache Storm (source: https://www.tutorialspoint.com/apache_storm/images/zookeeper_framework.jpg)

Queries in context of Storm are called topologies, which are defined as "directed graphs where the vertices represent computation and the edges represent the data flow between computation components" [2]. There are two types of vertices in topology, namely, spout and bolt. The former represents source of data tuples used within topology and the latter is held responsible for processing the data. Spouts can be written to read data from sources of data such as database, messaging frameworks, distributed file systems, etc. Bolts can perform simple transformations for streams, and complex stream transformations require multiple bolts. Storm mainly has two main types of nodes, namely Nimbus (master node) and Supervisor (worker node). The former is the central component of Storm and its main job is to run the topology. It analyzes and gathers the tasks which are to be performed and distribute these tasks to the supervisors available. Supervisors may have one or many worker processors. They assign the tasks to worker processors. Their main job is to spawn worker processors based on the instructions it is getting from Nimbus. Storm utilizes an internal centralized message network for coordination in between Nimbus and Supervisors, which is named as Zookeepers. Zookeeper. Along with creating worker processors, supervisors also monitor those worker processors and regenerate workers as necessary. Storm guarantees that every message is processed at-least once [1].

2.2 Apache Flink

Apache Flink is an open-source real-time processing framework which is used for processing streaming data. It is used for scalable, high performance and accurate real-time applications. Similar to the Apache Storm, Flink uses master-worker model. It consists of two element types, namely Job manager, which acts as the master and one or more Task Manager(s), which act as the worker(s). Unlike Storm, there is no layer in between the master and the workers like Zookeeper in Storm. Job Manager is held responsible for creating the execution graphs after receiving the job dataflow graphs from the client. It is also responsible for assigning the jobs to Task Managers within the clusters and supervising the execution of the jobs. Each Task Manager is held responsible for executing the tasks assigned to it by the Job Manager. Each Task Manager has a certain number of cluster computing slots used for parallelizing the tasks. The slot number can be configured and it is recommended to use as many slots as the number of CPU cores present in each Task Manager node. They should also send the status of execution of tasks to the Job Manager. Flink guarantees that every message is processed exactly once. Apache Flink is usually written in Java and Scala.

As compared to Apache Storm relative to the performance, Flink has very low latency and high throughput [5].

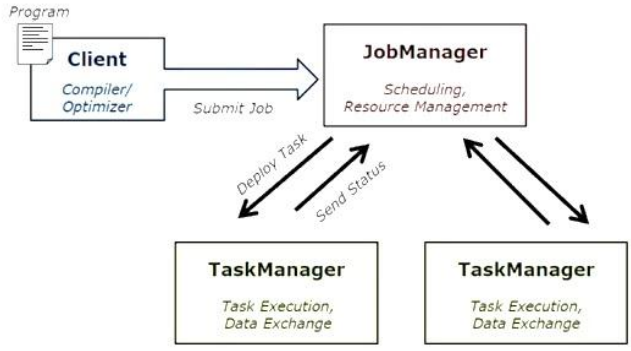

Fig-1: Apache Flink architecture (source: https://www.tutorialspoint.com/apache_flink/images/execution_architecture.jpg)

2.3 Summarized comparison of the frameworks

Framework/Properties	Apache Storm	Apache Flink
Source model	Open source	Open source
Processing model	Stream	Stream
Programming languages used in writing	Java, Clojure	Java Scala
Processing done for messages	At least once	Exactly once
Throughput	Low	High
Latency	Low	Very low
Important System components	Nimbus, Zookeeper, Supervisor	Job Manager, Task Manager

3. CONCLUSION

In this paper, the state-of-art technologies in the area of data stream processing, along with the evolution of data streaming processes have been discussed. This evolution discusses how the data streaming came into existence from batch processing. Also, two important frameworks in the field of data stream processing in Big Data, Apache Storm and Apache Flink have been discussed. A summarized comparison among the two frameworks has been given. The two frameworks are used for data stream processing in real-time applications.

REFERENCES

[1] Hesse, G., & Lorenz, M. (2015). Conceptual Survey on Data Stream Processing Systems. 2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS). doi: 10.1109/icpads.2015.106

[2] Toshniwal, A., Donham, J., Bhagat, N., Mittal, S., Ryaboy, D., Taneja, S.,... Fu, M. (2014). Storm@twitter. Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data - SIGMOD 14. doi: 10.1145/2588555.2595641

[3] Dean, J., & Ghemawat, S. (2008). MapReduce. Communications of the ACM, 51(1), 107–113. doi: 10.1145/1327452.1327492

[4] Real-time Data Stream Processing - Challenges and Perspectives. (2017). International Journal of Computer Science Issues, 14(5), 6–12. doi: 10.20943/01201705.612

[5] Kolajo, T., Daramola, O., & Adebiyi, A. (2019). Big data stream analysis: a systematic literature review. Journal of Big Data, 6(1). doi: 10.1186/s40537-019-0210-7

[6] Krettek, A. (2019, February 28). The data processing evolution: A potted history. Retrieved from https://www.itproportal.com/features/the-data-processing-evolution-a-potted-history/

[7] PrakashAll, C. (n.d.). Apache Storm: Introduction. Retrieved from https://www.linkedin.com/pulse/apache-storm-introduction-chandan-prakash

[8] Tiwari, A. (2017, November 17). Apache Storm: The Hadoop of Real-Time - DZone Big Data. Retrieved from https://dzone.com/articles/apache-storm-the-hadoop-of-real-time-1