ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

IJESRT
INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

COMPARISON OF MICRO-BATCH AND STREAMING ENGINE ON REAL TIME DATA

Dilraj Kaur*, Dr. Raman Chadha, Nitin Verma
* Research Scholar, M.Tech(CSE),CGC Technical Campus, Jhanjeri
Professor, HOD(CSE), CGC Technical Campus, Jhanjeri
A.P(CSE), CGC Technical Campus, Jhanjeri

DOI: 10.5281/zenodo.569956

ABSTRACT

Big Data analytics has recently gained increasing popularity as a tool to process large amounts of data on-demand. Spark and Flink are two Apache-hosted data analytics frameworks that facilitate the development of multi-step data pipelines using directly acyclic graph patterns. Making the most out of these frameworks is challenging because efficient executions strongly rely on complex parameter configurations and on an in-depth understanding of the underlying architectural choices. Although extensive research has been devoted to improving and evaluating the performance of such analytics frameworks, most of them benchmark the platforms against Hadoop, as a baseline, a rather unfair comparison considering the fundamentally different design principles. This paper aims to bring some justice in this respect, by directly evaluating the performance of Spark and Flink. Our goal is to identify and explain the impact of the different architectural choices and the parameter configurations on the perceived end-to-end performance. To compare the performance of Flink and Spark streaming using E-commerce data. Flink and Spark are both general-purpose data processing platforms and top level projects of the Apache Software Foundation (ASF). They have a wide field of application and are usable for dozens of big data scenarios.

INTRODUCTION

As the huge amount of data is generated every day, It creates a lots of challenges to find new ways to handle huge data effectively those challenges are-

- Data capturing
- Data storage
- Querying and Analyzing data

In past several system were developed to processes big data. Most of them were based on MapReduce and spark framework. There is some drawback of these framework so some new framework were developed which has made querying and analyzing data at a large scale much more efficient than previous frameworks. These are design for distributed processing of large data sets across clusters of computers. Apache spark and Apache Flink both are open source platform for batch processing as well as streaming processing engine at massive scale which provides fault-tolerance and data-distribution for distributed computations.

EXISTING SYSTEM - SPARK

Spark is an open source data processing framework, which was developed in 2009 and open sourced in 2010 as an Apache Project. Spark is a lightning and faster cluster computing technology. Spark support “in-memory computation” ,which becomes advantage as compared to other big data technologies like hadoop and storm. Spark has written in Scala language. It has some code written in Java ,Python and R.Spark helps to simplify the challenges and compute intensive task of processing high volume of real time or archived data both structured or unstructured. Spark integrating relevant complex capabilities like Machine learning, and Graph algorithms. User can combine all these capabilities seamlessly in single workflow.

WHAT IS SPARK STREAMING?

Spark streaming operates on the concept of micro-batches. This means that Spark Streaming should not be considered a real-time stream processing engine. This is perhaps the single biggest difference between Spark

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

Streaming and other platforms such as Apache Storm or Apache Flink. Spark Streaming receives live input data streams and divides the data into batches, which are then processed by the Spark engine to generate the final stream of results in batches.

LIMITATION OF SPARK

The biggest Limitation about running spark stream in production are **back pressure, batch size, state management** and **out of order data.**

- **Back pressure** occurs when the volume of events coming across a stream is more than the stream processing engine can handle. There are changes that will show up in version 1.5 of spark to enable more dynamic ingestion rate capabilities to make back pressure be loss of an issue.
- **Out of order data:** More work is being performed to enable user - defined time extraction function. This will enable developer to check event time against event already processed.
- **Batch Size:** Spark streaming needs batch size to be defined before any stream processing. It's because spark streaming follows micro batches for stream processing which is also known as near realtime .
- **State Management: In** spark, after each batch, the state has to be updated explicitly if you want to keep track of word count across batches.

NEW SYSTEM - FLINK

Apache Flink is a true stream processing tool. Flink's core is a streaming dataflow engine which also provides distributed processing, fault tolerance, etc. Flink is a top level project of Apache. Flink is a scalable data analytics framework that is fully compatible to Hadoop. Flink can execute both stream processing and batch processing easily. Flink is an alternative of Mapreduce and Spark framework. Its 100 times faster than other framework. Flink is not dependent of hadoop but it uses hdfs to read and write data. Flink does not provide its own data storage system.it takes data from distributed storage.

WHAT IS FLINKSTREAMING?

Flink Streaming uses the pipelined Flink engine to process data streams in real time and offers a new API including definition of flexible windows.

Advantages of Flink Streaming

Apache flink reduces the complexity that has been faced by other distributed data driven engines. it is achieved by integrating query optimization, concepts from database systems and efficient parallel in-memory andout-of-core algorithms, with the MapReduce framework.

- **No need to Batch Size in Flink :**Spark streaming needs batch size to be defined before any stream processing. It's because spark streaming follows micro batches for stream processing which is also known as near real-time . But flink follows one message at a time way where each message is processed as and when it arrives. So flink does not need any batch size to be specified.
- **State Management: In** spark, after each batch, the state has to be updated explicitly if you want to keep track of word count across batches. But in flink the state is up-to-dated as and when new records arrive implicitly.
- **Support for Event Time and Out-Of-Order Events :**Flink supports stream processing and windowing with **Event Time** semantics.
    - Event time makes it easy to compute over streams where events arrive out of order, and where events may arrive delayed.

COMPARISON

Features	**Apache Flink**	**Apache Spark**

Computation Model	Flink is based on operator-based computational model.	Spark is based on micro-batch modal.
Streaming engine	Apache Flink uses streams for all workloads: streaming, SQL, micro-batch and batch. Batch is a finite set of streamed data.	Spark uses micro-batches for all workloads. But it is not sufficient for use cases where we need to process large streams of live data and provide results in real time
Iterative processing	Flink API provides two dedicated iterations operation Iterate and Delta Iterate.	Spark is based on non-native iteration which is implemented as regular for – loops outside the system.
Optimization	Apache Flink comes with an optimizer that is independent with actual programming interface.	In Apache Spark jobs has to be manually optimized.
Latency	With minimum efforts in configuration Apache Flink's data streaming run-time achieves low latency and high throughput.	Apache Spark has high latency as compared to Apache Flink.
Performance	Overall performance of Apache Flink is excellent as compared to any other data processing system. Apache Flink uses native closed loop iterations operators which makes machine learning and graph processing more faster.	Though Apache Spark has an excellent community background and now It is considered as most matured community. But Its stream processing is not much efficient than Apache Flink as it uses micro-batch processing.
Fault tolerance	The fault tolerance mechanism followed by Apache Flink is based on Chandy-Lamport distributed snapshots. The mechanism is lightweight, which results in maintaining high throughput rates and provide strong consistency guarantees at the same time.	Spark Streaming recovers lost work and delivers exactly-once semantics out of the box with no extra code or configuration.
Duplicate elimination	Apache Flink process every records exactly one time hence eliminates duplication.	Spark also process every records exactly one time hence eliminates duplication.
Window Criteria	Flink has a record-based or any custom user-defined Window criteria.	Spark has a time-based Window criteria
Memory -Management	Flink provides automatic memory management.	Spark provides configurable memory management. Spark 1.6, Spark has moved towards automating memory management as well.

INFRASTRUCTURE STATISTICS

- Number of nodes in the cluster: 1 Node
- Node Configuration: Dual-core Processor, 4 GB RAM
- spark-2.0.0-bin-hadoop2.7
- flink-1.2.0-bin-hadoop26-scala_2.10

Data Statistics

- Data is in the form of JSON.
- Each record have fixed number of fields.
- Average record size is 3000 Bytes

Performance of Flink Streaming Vs Spark streaming

Query \ No Of Record->	**1000**	**5000**	**10000**	**50000**	**100000**	**500000**
Monthly distribution of reviews	Flink-125 Spark-51	Flink-134 Spark-63	Flink-142 Spark-54	Flink-178 Spark-57	Flink-207 Spark-48	Flink-244 Spark-132

Monthly average ratings of new amazon reviews	Flink-103 Spark-63	Flink-110 Spark-74	Flink-116 Spark-80	Flink-123 Spark-61	Flink-188 Spark-53	Flink-193 Spark-56
Product with highest no of reviews	Flink-274 Spark-45	Flink-365 Spark-50	Flink-398 Spark-61	Flink-428 Spark-46	Flink-582 Spark-44	Flink-543 Spark-68
Distribution of rating of products (rating from 1-2, 2-3, 3-4, 4-5)	Flink-75 Spark-60	Flink-86 Spark-48	Flink-94 Spark-53	Flink-96 Spark-65	Flink-104 Spark-100	Flink-85 Spark-69
Distribution of helpfulness of reviews (% of helpfulness)	Flink-134 Spark-45	Flink-232 Spark-50	Flink-184 Spark-61	Flink-526 Spark-46	Flink-610 Spark-44	Flink-543 Spark-68

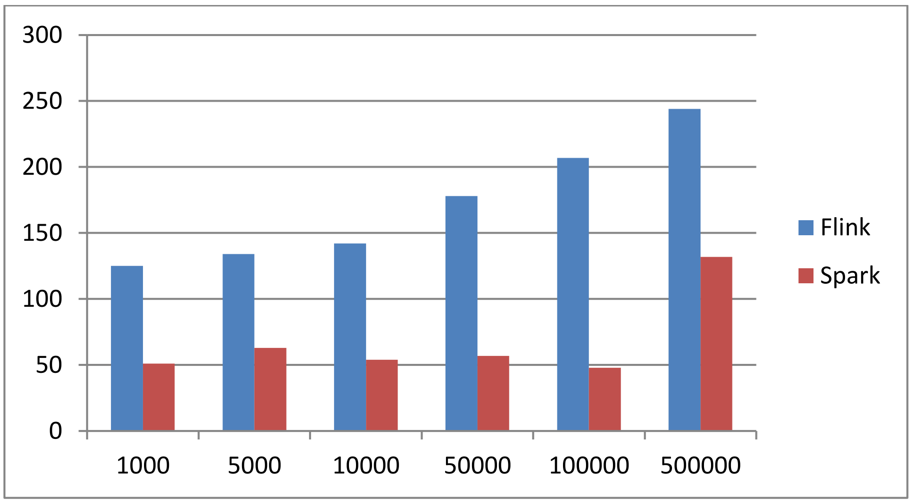

Fig 1: Monthly distribution of reviews on Amazon Data

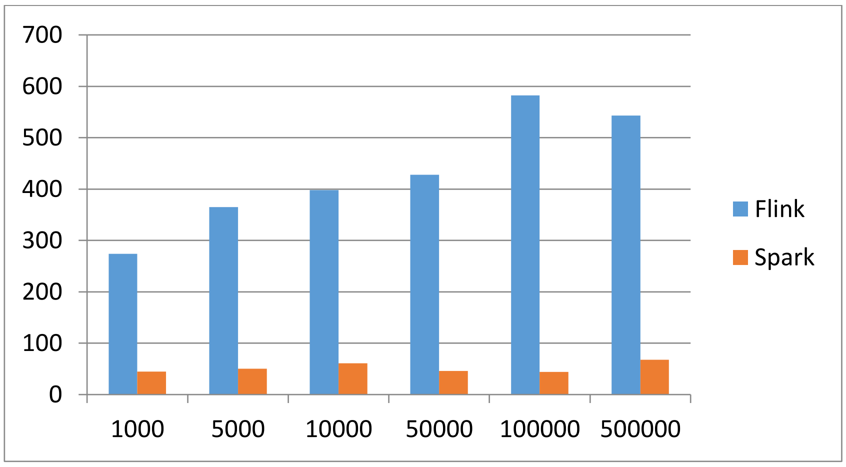

Fig 2: Monthly average ratings of new Amazon reviews

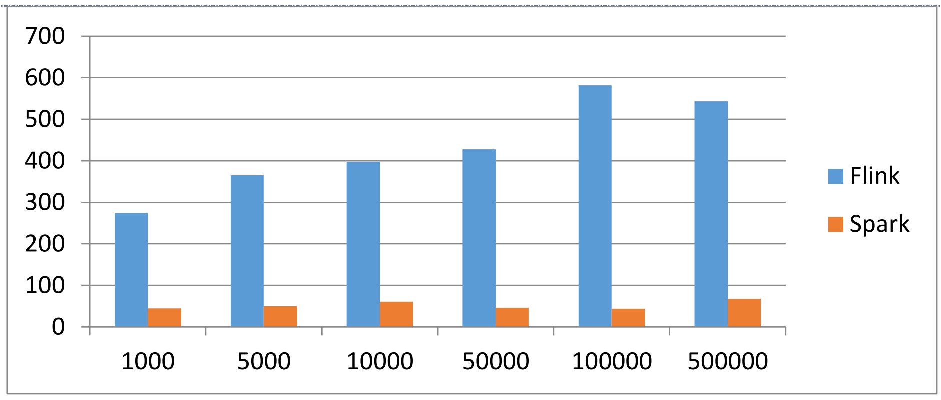

Fig 3: Product with highest no of reviews

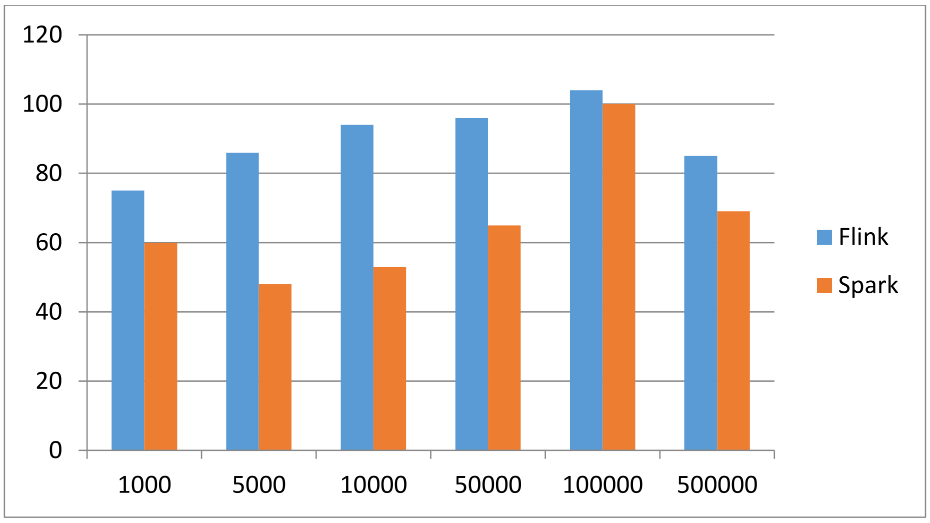

Fig 4: Distribution of rating of products (rating from 1-2, 2-3, 3-4, 4-5)

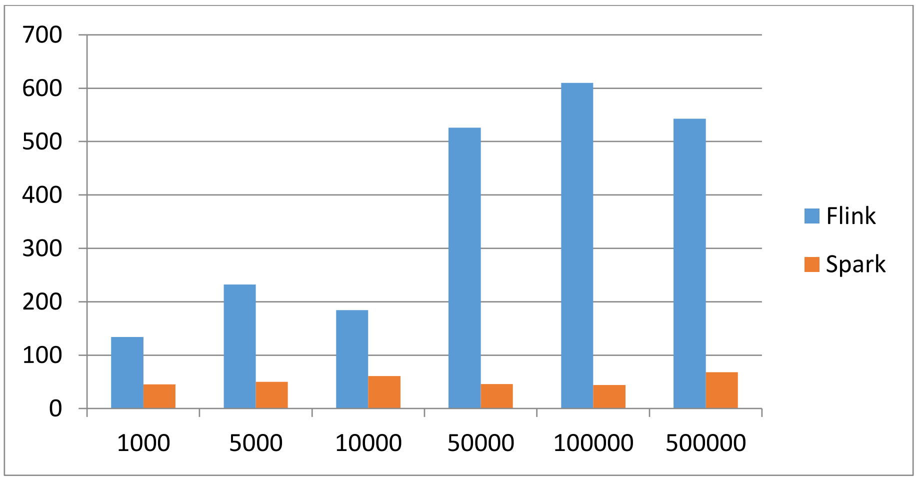

Fig 5: Distribution of helpfulness of reviews (% of helpfulness)

CONCLUSION

Apache Spark and Flink both are next generation Big Data tool grabbing industry attention. Both provide native connectivity with Hadoop and NoSQL Databases and can process hdfs data. Both are nice solution to several Big Data problems. But Flink is faster then Spark, due to its underlying architecture. Apache Spark is most active component in Apache repository. Spark has very strong community support and has good number of contributors. Spark has already been deployed in the production. in our experiment we use amazon data ,average time to process data with flink is 240.3sec and spark is 60.4sec and the performance of spark is 179.5% better than over flink.

REFERENCES

[1] Tom White, "Hadoop Definitive Guide". O'Reilly
[2] Asterios Katsifodimos(2016) et. al. "Apache Flink: Stream Analytics at Scale"
[3] Altti Ilari Maarala (2015)"Low latency analytics for streaming traffic data with Apache Spark" IEEE International Conference.
[4] Kuo M.H., Sahama T., Kushniruk A.W., Borycki E.M., Grunwell D. Health Big Data Analytics: Current Perspectives, Challenges and Potential Solutions. Int J Big Data Intelligence 2014; 1(12): 114–126.
[5] Sun J., Reddy C.K. Big Data Analytics for Healthcare. Tutorial presentation at the SIAM International Conference on Data Mining, Austin, TX, 2013.
[6] Nelson, R., Staggers, N. Health Informatics: an interprofessional approach. Mosby, an imprint of Elsevier Inc.; 2014. Saint Louis, MO.
[7] Moselle, K. Data Management in the Island Health Secure Research Environment. Enterprise Architecture at Vancouver Island Health Authority. Working Draft 5; 2015. Victoria, BC.
[8] Hadoopproject. http://hadoop.apache.org/.
[9] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wycko, and R. Murthy., "Hive A Warehousing Solution Over a MapReduce Framework." VLDB, 2009.
[10] R. Stewart. Performance and Programmability of High Level Data Parallel Processing Languages: Pig, Hive, JAQL & JavaMapReduce, 2010. Heriot-Watt University.
[11] Y. Jia and Z. Shao. A Benchmark for Hive, PIG and Hadoop,2009 https://issues.apache.org/jira/browse/HIVE
[12] Rasim Alguliyev and Yadigar Imamverdiyev. Big data: Big promises for information security.In Application of Information and Communication Technologies (AICT), 2014 IEEE 8th Interntional Conference on, pages IEEE, 2014
[13] Apachehadoop en.wikipedia.org/wiki/apache hadoop.
[14] Sam Madden. From databases to big data. IEEE Internet Computing, 16(3):4{6, 2012.
[15] https://www.quora.com/What-are-the-differences-between-batch-processing-and-stream-processing-systems.

CITE A JOURNAL:

Kaur, D., Chadha, R., & Verma, N. (2017). COMPARISON OF MICRO-BATCH AND STREARMING ENGINE ON REAL TIME DATA. ***INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY,*** ***6*****(4), 756-761. doi:10.5281/zenodo.569956**