Research on Advanced Streaming Processing on Apache Spark

ABSTRACT

Today's digital world computations are tremendously difficult and they always demand essential requirements to significantly process and store datasets of enormous size for a wide variety of applications. Since the volume of digital world data is enormous, unstructured data are mostly generated at high velocity beyond limits and are doubled day by day. Over the last decade, many organizations have been facing major problems in handling and processing massive chunks of data, which could not be processed efficiently due to lack of enhancements on existing and conventional technologies. This paper addresses how to overcome these problems efficiently using the most recent and world primary powerful data processing tool, namely clean open-source Hadoop, one of its core components being Map Reduce that is subject to few performance issues. The objective of this paper is to address and overcome the limitations and weaknesses of Map Reduce with Apache Spark.

KEYWORDS: *Big data; Hadoop; HDFS; Map reduce; Apache spark; Processing.*

1. Introduction

Today's digital world has significantly enhanced emerging technologies, new mechanisms, techniques, and tools and media such as social networks that generate huge data whose size grows exponentially each year. The IDC research forecasts that the total volume of data could increase 50 times by 2020 and they are mostly driven by different types of sensors, medical equipment, banking, Twitter, Facebook, Google, etc. Ninety percent of the last-decade applications have generated unstructured data such as documents, mail, images, etc. Nearly three Twitter messages are generated per minute for 26,976 years [1]. In this regard, the overall number of servers running data storage worldwide could increase ten-fold during the next decade. Big data become hot topics in the present decade, which seems to be an enormous size, and a majority of the unstructured data can, typically, never be processed using conventional computational tools and methods in an efficient, scalable, cost-effective, and reliable manner. In the digital world, the term big data in the latest era is characterized by immensely structured, semi-unstructured data sets that are handled by conventional data processing tools and methods; however, they have insufficient and poor performance.

Big data are significantly supported to efficiently analyze in-depth concepts to facilitate better intelligent and strategic decision-making for the development of the organization. Big data are distinctly addressed in terms of six characteristics: V's volume, velocity, veracity, variety, value, variability, and validity [2].

2. Background Challenges

Major limitations of existing approaches result from the capacity to only process low-volume datasets, which could be hosted only by standardized servers of the database or until the data processor is limited. However, it is an exhausting environment to process such information using a single database capacity, especially for managing massive amounts of extensible data. Map Reduce needs a longer time to perform maps and reduce tasks, thereby raising the latency. Reducing data processing velocity and increasing delay in distributing and processing the large data are given in the following clusters in Map Reduce [3].

*

Major problem: real-time computation is required; therefore, it is more essential to complete a task without delaying the next step or actual deadline of completion.

Critical path issue: Internet world computations demand a real-time manner to complete each job without halting the next step or delaying the exact deadline of completion. Therefore, if one computer slows down the actual work, all other works will be delayed.

Problem of reliability: Typically, every machine works with a portion of the data and failing to do so is a major challenge.

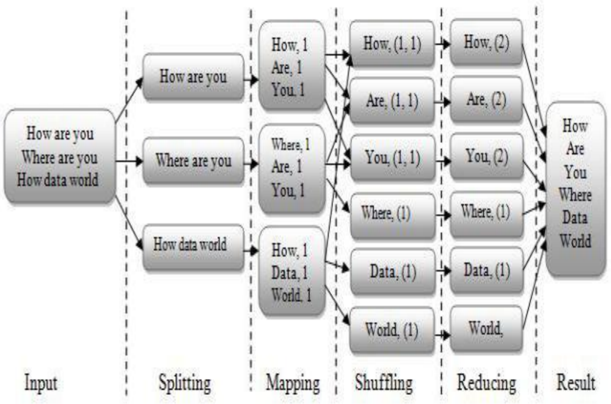
Equal splitting problem: the data can be split into slightly smaller pieces in order for every computer to use a portion of the data. This means that the data can be split up in such a way that no single machine gets overwhelmed or overused.

Possible failure of single splitting: we could not calculate the outcome when a device fails to work to produce the total output. A framework should guarantee that the systems would achieve a more fault-tolerant ability.

Result aggregation: This mechanism could be applied to aggregate the outcome produced from the final output by each device since the difficulties should be dealt with separately while only processing massive data sets in parallel using conventional methods [4].

3. Map Reduce

Google has introduced the latest tool to significantly solve existing technological difficulties by Map Reduce which is a simple, open-source, more distributed, and parallel processing framework. This is the first and foremost latest digital world framework that is widely used to process massive volumes of world data on a large commodity cluster with reliability, high fault tolerance, and great scalability. In December 2004, Google issued a journal on Map Reduce. The great benefit of Map Reduce is that multiple computer nodes make it easy to modify data processing [5]. Apache Hadoop Map Reduce has been used extensively over the past decade for parallel computing of massive-size data; this is becoming a de-facto benchmark in these areas. Map Reduce performs two different tasks, Map and Reduce, which take place entirely after the completion of the mapped phase. The map task actually reads and processes a data block at an intermediate level of outputs so as to produce key value pairs. The reducer receives from several map jobs the key value pair and then, adds a smaller set of multiples or key-value pairs (the final output) by means of intermediate datasets (intermediate key-value pair), as shown in Fig. 2.



g. 1. MpaReduce word counting

Features of map reduce

Parallel Processing: Typically split the job into several nodes on Map Reduce; moreover, every node operates in parallel to other parts of the job. Map Reduce follows the divide-and-conquer strategy, which enables us to manipulate data with multiple machines [6]. Actually, the time taken to meticulously process the data is significantly reduced by a number of machines rather than one machine in parallel.

Tab. 1. Map phase with key and values

	Input	Output
Map Phase	<key1, value1>	list (<key2, value2>)
Reduce phase	<key2, list(value2)>	list (<key3, value3>)

Fault Tolerance: Hadoop carefully controls the faults using replication factor. Whenever user stores a particular file in hadoop storage component HDFS, the file is partitioned into a number of blocks and data blocks are distributed over various machines in HDFS cluster. In addition, the default replica value 3 in each block is generated on other cluster machines. If one machine in the cluster fails during critical circumstances, the user could gain data from other machines [7].

Scalability: Typically, one of the major strengths of Hadoop is scalability. Therefore, new nodes are very simply added with no downtime. Hadoop supports horizontal scalability; therefore, the latest nodes are added in a fly manner to the machine. In Apache Hadoop, every application can run significantly on more than thousands of nodes.

Reliability: In Hadoop, the whole data become more reliable which are stored on the cluster of machines regardless of machine failure because replication mechanism can support gaining the same data from different places. Therefore, if any of the nodes fails, then also we can store data reliably.

High Availability: given a large number of copies of datasets, actual data become more available and easy access despite hardware failures. Therefore, any device goes down, but our required data could be retrieved another way.

Data locality: The major limitation of Hadoop is that it is subject to much crossing-switching system traffic from the enormous quantity of data. Therefore, to overcome this problem, data locality has been fabricated. Hadoop can facilitate the computation which is very closely tied to real data and it actually resides on the cluster node. Therefore, network congestion is extremely reduced, but system throughput is enlarged.

The algorithm employed to count each word showing up in a set of documents:

```
Step1: function map task (String name, String document):
Step2: for each word w in a document:
Step3: emit (w, 1)
Step4: function reduce task (String word, Iterates partial Counts):
Step5: sum = 0
Step6: for each pc in partial Counts:
Step7: sum += pc
Step8: emit (word, sum)
```

There is a lack of real-time processing because it could not continuously use every aspect by Map Reduce. However, the intermediary processes are mostly required to respond to each other (each job runs in isolation). Typically, processing requiring huge data could be shuffled across the network. In some cases, handling the streaming process using Map Reduce is highly difficult, because it is highly suitable to batch process enormous amounts of data [8].

Some major issues associated with Hadoop: Hadoop is subject to a number of drawbacks such as small files, processing speed very low, high latency, less security, poor real-time stream processing, efficient support up to batch processing only, more uncertainty, complex line of code, no mechanism of caching, complicated application, generally vulnerable, lack of delta iterations, poor interactive processing, and lack of in-memory and graph processing [9]. Plus, its programming interface is poor. Through the continually rising actual size of data-disseminated programming, the existing models including Map Reduce and its open-source application Hadoop may face performance-wise difficulties. To this end, Apache Spark was employed to address the problems and drawbacks of Map Reduce.

4. Apache Spark

Apache Spark is really the fastest general-purpose cluster-computing model, which is more distributed, parallel, and completely open-source. It was originally developed in 2009 and revealed in 2010 as an Apache project in the UC Berkeley's AMPLab [10, 11, 12]. Spark offers more interface mostly with underlying data parallelization and fault tolerance for programming on whole clusters. Spark has significantly proven incredibly popular and more widely used by many real-world large corporations for enormous, multi-peta to zettabyte data storage and analysis computations due to its lighting fast and in-memory processing. Last year, Apache Spark implements a benchmark examination by sorting 100 terabytes of data in 23 minutes and the previous world record of 71 minutes could be held by Hadoop. Spark Core, which is at the center of a project,

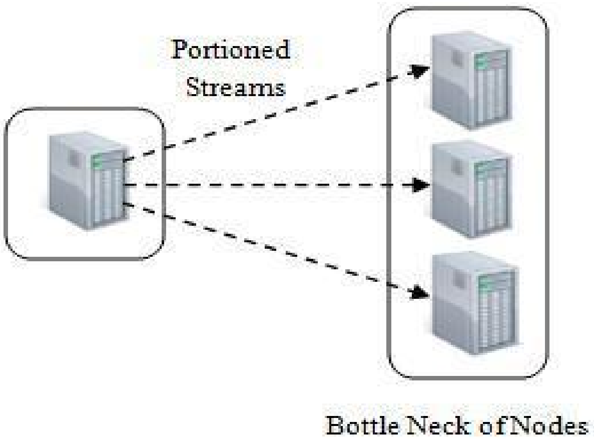
provides decentralized functionality, programming, and transmission and offers programmers a significantly faster and flexible effective alternative to Map Reduce. Developers interested in Spark conclude that when a massive volume of data is properly processed using the memory approach, it can work effectively and extremely (lighting) faster, meaning 100 times faster than Map Reduce and also 10 times faster than disk. Spark simply provides greater scalability, higher fault tolerance, reliability, and several other features [13, 14, 15].

Spark and its RDDs was revealed in 2012 and the response to the drawbacks of the Map Reduce

International Journal of Industrial Engineering & Production Research, March 2021, Vol. 32, No. 1

cluster computing model makes a selective straight dataflow construction on distributed applications; Map Reduce applications gather input data in the disk source and, then, claim responsibility over the data, degrade the outcome of the map task, and tremendously store reduce task outcome on disk. Spark's RDDs could perform similarly to a working set of distributed applications which contribute to a (intentionally) reduced pattern of an oriented distributed shared memory. Apache Spark enjoys complete service of architectural establishment with Resilient Distributed Dataset (RDD), thus supporting extremely read-only operation on the number of data items. Circulation across the cluster is significantly implemented in a fault-tolerant manner.

The Data frame API could be generated as a notion on top of the RDD. In apache, Spark is a starting interface which is an Application-Programming Interface (API) and is called the RDD. Spark strongly supports memory processing to enhance the performance of applications for big data analysis, but it could also execute traditional disk-based treatment whenever datasets are far too large for the system memory available. The RDD is specifically designed so that customers can cover up a great deal of computational complexity. It aggregates data and partitions them across a whole server cluster where it could be calculated, migrated, or simply run via an analytical paradigm in another data store [16].



onventional System with Static Scheduling

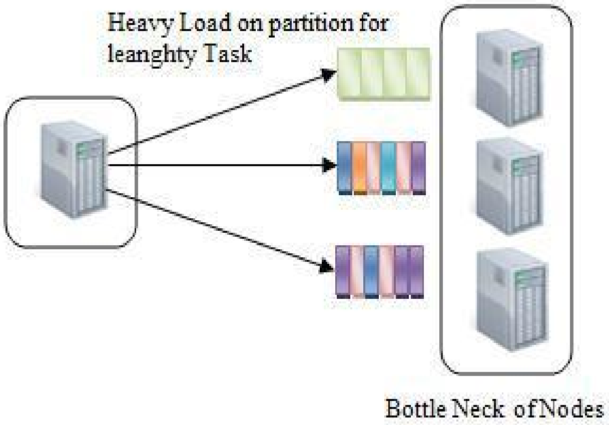


Fig. 3. Spark Streaming with Dynamic Scheduling

The conventional approach has been utilized by many systems, but distributions are more computationally exhaustive than the others and the nodes are assigned statically to processes that are partitioned based on batch processing; however, a major introduced issue is a bottleneck that slows down the entire process. Overcoming this issue for using Spark streaming is shown in Figs. 1, 2. The most significant benefit of Apache Spark is its high speed. Digital world applications could operate 100X active in-memory and 10X quicker when working on a disk drive. Compared with Hadoop, Spark keeps the intermediate outcomes in memory (rather than disk). Spark significantly reduces the number of disk I/Oís. Fig. 4 reveals a flash of the logistic regression in Hadoop and Spark and they are starkly distinct in terms of operating times.

Spark greatly supports the significant application of iterative-based techniques, which handles different times for datasets with a loop service, especially data analysis with a complete interactive manner. Spark ensures the low latency and applications could be degraded at different dimensions, compared with Apache Hadoop Map Reduce execution. Among the many types of iterative techniques are the training-based methods for machine learning operations, making a fundamental incentive for improving Apache Spark.

It can easily handle large batch and real-time real-world analytics, significant data processing tasks and interactive queries, and machine learning. Apache Spark accomplishes higher performance for both batch and streaming information using a state-of-the-art DAG scheduler, a query optimizer, and a psychical execution engine. Apache Spark always needs a cluster-based administrator as well as the latest parallel and more distributed storage segment. Through batch control, spark recommends standalone mode, Hadoop YARN, or Apache Mesos. In the propagated accommodation, Spark makes interface with extensive diversity including Hadoop Distributed File System (HDFS), and a custom solution could be executed. Spark further establishes a pseudo-distributed restricted mode and is normally utilized for improvement or examination objectives, where disseminated accommodation is not required and the local file operation can be utilized alternately in such a situation. Spark operates on a single machine including one executor through one CPU kernel.

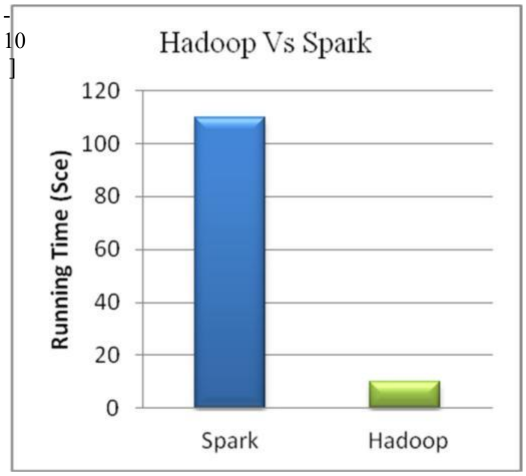


Fig. 4. Hadoop and Spark Running Times

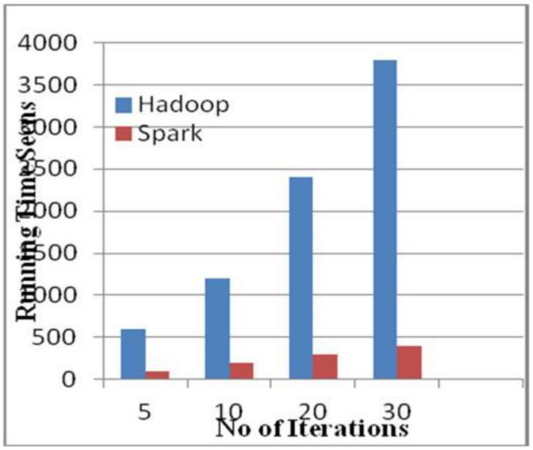


Fig. 5. Hadoop and Spark Running Times

International Journal of Industrial Engineering & Production Research, March 2021, Vol. 32, No. 1

Spark libraries

The Spark Core engine processes mainly provide high-level API and support a similar set of related data management and analysis tools. In addition to spark core, a new package of the most popular code libraries to be used in data analysis software programs comes with an apache spark environment. Spark SQL allows users to query the stored data in different applications in the relevant SQL language [10, 12]. Spark streaming can simply build an application to evaluate and present information even in real-time.

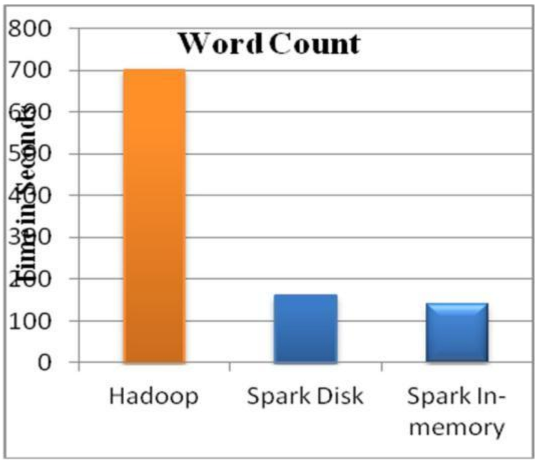
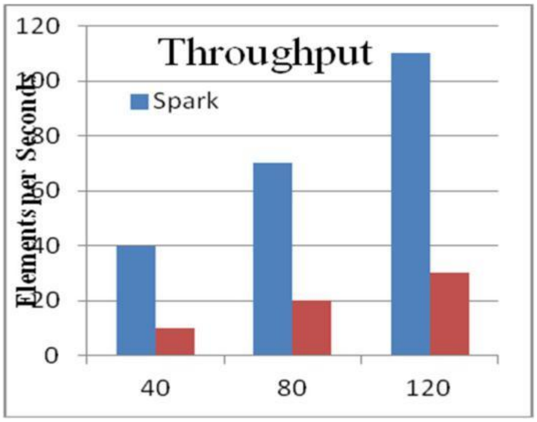


Fig. 6. Hadoop and spark Word Count Times



7. Hadoop and spark Throughput

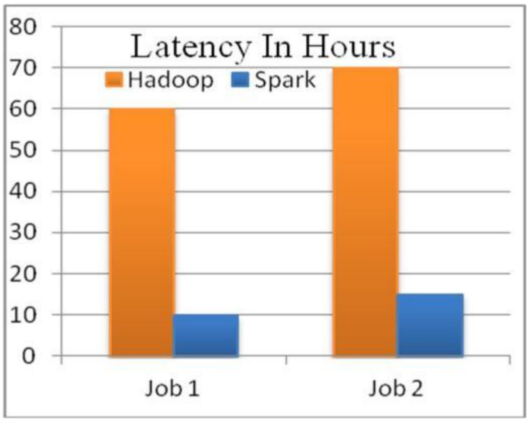


Fig. 8. Hadoop and spark latency

MLlib is a device that attempts to learn a code library that allows customers to use advanced mathematical operations on spark information and create new applications for those analyses. GraphX which is a graph-parallel numerical computation online tool is a built-in library.

Spark typically provides more than 80 operators which simply make parallel applications easy to develop. In contrast to Scala, Python, R, and SQL shells, one could use the program interactively. Spark enables a pack of libraries along with SQL and Data Frames, MLlib, GraphX, and Spark Streaming. In almost the same application, you can dynamically incorporate these libraries. Then, one can continue to run Spark mostly on EC2, Hadoop YARN, Mesos, or Kubernetes by using its independent cluster mode [14, 16, 17,18, 19].

Tab. 2. Sample document with Customer Tweets

	Hadoop Map Reduce	Spark
Processing	Batch	Micro Batch, Stream
Speed	Slow	Faster than MR
Operators	NA	Time-based
Windows	No	Yes
Storage data	HDFS	In-Memory
Latency	High	Low
Fault tolerance	High	High, RDD DAG
Performance	Slow	High than MR
Removal of duplicate	High	Process records exactly
Iterative data flow	Chain of states	Cyclic data flow DAG
Scalability	Incredible up to 10,000	High cluster of 8000
Visualization	Low	High, need RAM
Recovery	high fault tolerant	RDD DAG
Abstraction	NO	Spark RDD Data stream
Easy to use	Difficult	Easy
Realtime analysis	No	Good
Scheduler	Fair, capacity	Own flow scheduler
SQL	Hive	SSQL Hive, FDSL
Catching	Not	Yes
Hardware	Commodity h/w	Mid to high level h/w
Machine learning	Mahout	Mlib
Line of code	1,20,000	20,000
Deployment	Fully distribute mode	Standalone on mesos/YARN

Hive and hundreds and hundreds of many other sources of data have access to much data. They are characterized by swift processing, natural dynamism, greater in-memory computation in spark, reusability, fault tolerance in spark, real-time stream processing, poor evaluation in apache-spark, supporting multiple languages, active, progressive and expanding spark community, support for sophisticated analysis, integrated with Hadoop, Spark GraphX, cost-effective manner.

5. Citations

Today's digital world follows the most centralized analytics engine for large-scale data processing. The current internet-based applications enjoy higher speed, greater storage, and stream processing for a massive volume of data in a reliable manner in the Apache Spark framework. The main objective of Spark is to overcome the performance issues as well as limitations of Map Reduce. It managed to provide good latency, throughput, fast execution, and stream processing.