© 2017 IJEDR | Volume 5, Issue 2 | ISSN: 2321-9939

Big Data Computing Framework: A Compact Review

R K Jena,
Institute of Management Technology, Nagpur

***Abstract*-Advances in information technology and its widespread growth in several areas of business, engineering,medical, and scientific studies are resulting in information/data explosion. Knowledge discovery anddecision-making from such rapidly growing voluminous data are a challenging task in terms of data organizationand processing, which is an emerging trend known as big data computing, a new paradigm thatcombines large-scale compute, new data-intensive techniques, and mathematical models to build data analytics.Big data computing demands a huge storage and computing for data curation and processing thatcould be delivered from on-premise or clouds infrastructures. Therefore a newrobust architecture is needed to store and process information. Initially new programing approaches developed based on parallel processing and then distributed computing programing models were developed. This is where Hadoop, Spark,Flinkand Storm frameworks are developed and deployed for big data processing since last few years. These platforms are showing promising results.This paper discusses the evolution of big data computing platforms and compare Hadoop, Spark, Flink and Storm framework.**

***Index Terms*-Words: Big Data, Hadoop, Spark, Flink andStorm**

I. INTRODUCTION

Big data computing is an emerging data science paradigm of multidimensional information mining for scientific discovery and business analytics over large-scale infrastructure. The data collected/produced from several scientific explorations and business transactions often require tools to facilitate efficient data management, analysis, validation, visualization, and dissemination, while preserving the intrinsic value of the data [1–5]. The IDC [6] report predicted that there couldbe an increase of the digital data by 40 times from 2012 to 2020. New advancements in semiconductor technologies are eventually leading to faster computing, large-scale storage, and faster and powerful networks at lower prices, enabling large volumes of data preservation and utilization at faster rate. Recent advancements in cloud computing technologies are enabling to preserve every bit of the gathered and processed data, based on subscription models, providing high availability of storage and computation at affordable price. Conventional data warehousing systems are based on predetermined analytics over the abstracted data and employ cleansing and transforming into another database known as data marts; which are periodically updated with the similar type of rolled-up data. However, big data systems work on non-predetermined analytics; hence, no need of data cleansing and transformations procedures.

Big Data Applications

As business domains are growing, there is a need to converge a new economic system redefiningthe relationships among producers, distributors, and consumers of goods and services. Obviously, it is not feasible to depend on experience or pure intuition always; however, it is also essential to use critically important data sources for decision-making. The National Institute of Standards and Technology Big Data Public Working Group described a survey of big data architectures and framework from the industry [7]. The several areas of big data computing are described in the succeeding texts.

- **Scientific explorations**: The data collected from various sensors are analyzed to extract the useful information for societal benefits. For example, physics and astronomical experiments – a large number of scientists collaborating for designing, operating, and analyzing the products of sensor networks and detectors for scientific studies. Earth observation systems – information gathering and analytical approaches about earth’s physical, chemical, and biological systems via remote-sensing technologies – to improve social and economic well-being and its applications for weather forecasting, monitoring, and responding to natural disasters, climate change predictions, and so on.
- **Health care**: Healthcare organizations would like to predict the locations from where the diseases are spreading so as to prevent further spreading [8]. However, to predict exactly the origin of the disease would not be possible, until there is statistical data from several locations. In2009, when a new flu virus similar to H1N1 was spreading, Google has predicted this and published a paper in the scientific journal Nature [9], by looking at what people were searching for, on the Internet.

- **Governance**: Surveillance system analyzing and classifying streaming acoustic signals, transportation departments using real-time traffic data to predict traffic patterns, and update publictransportation schedules. Security departments analyzing images from aerial cameras, news feeds, and social networks or items of interest. Social program agencies gain a clearer understanding of beneficiaries and proper payments. Tax agencies identifying fraudsters and support investigation by analyzing complex identity information and tax returns. Sensor applications such stream air, water, and temperature data to support cleanup, fire prevention, and other programs.
- **Financial and business analytics**: Retaining customers and satisfying consumer expectations are among the most serious challenges facing financial institutions. Sentiment analysis and predictiveanalysis would play a key role in several fields like travel industry – for optimal cost estimations and retail industry – products targeted for potential customers. Forecast analysis – estimating the best price estimations and so on.
- **Web analytics**: Several websites are experiencing millions of unique visitors per day, in turn creating a large range of content. Increasingly, companies want to be able to mine this data to understand limitations of their sites, improve response time, offer more targeted ads, and so on. This requires tools to perform complicated analytics on data that far exceed the memory of a single machine or even in cluster of machines.

The big data challenges

Performing computation on big data is quite a big challenge. To work with huge volume of data that easily surpass several terabytes in size, requires distributing parts of data to several systems to handle in parallel. By doing it, the probability of failure rises. In a single single-system, failure is not something that usually program designers explicitly worry about [10]. However, in a distributed n scenario, partial failures are expected and common, but, if the rest of the distributed system is fine, it should be able to recover from the component failure or transient error condition and continue to make progress. Providing such resilience is a major software engineering challenge and concern [10].In addition, to these sorts of bugs and challenges, there is also the fact that the computer hardware has finite resources available. The major hardware restrictions include; Processor time, Memory, Hard drive space and Network bandwidth.

Individual systems usually have few gigabytes of memory. If the input dataset is set several terabytes, then this would require a thousand or more machines to hold it in RAM and even then, no single machine would be able to process or address all of the data. Hard drives are a lot bigger than RAM, and a single machine can currently hold multiple terabytes of information on its hard drives. But generated data of a large large-scale computation can easily require more space than what original data had occupied. During this, some of the storage devices, employed by the system may get full, and the distributed system will have to send the data to other node, to store the overflow. Finally, bandwidth is a limited resource. While a pack of nodes directly connected by a gigabit Ethernet generally experience high throughput between them, if all transmit multi-gigabyte, they would saturate the switch's bandwidth. In addition to that if the systems were spread across multiple racks, the bandwidth for the data transfer would be more diminished[10].

To achieve a successful large large-scale distributed system, the mentioned resourcesmust be efficiently managed. Furthermore, it must allocate some of these resourcestoward maintaining the system as a whole, while devoting as much time as possible tothe actual core computation [10].Synchronization between multiple systems remains the biggest challenge in distributedsystem design. If nodes in a distributed system can explicitly communicate with one another, then application designers must be cognizant of risks associatedwith such communication patterns. Finally, the ability to continue computation in the heface of failures becomes more challenging [1].

II. STATE OF THE ART DEVELOPMENTS

This section will begin to explain the up-to-date solutions for the big data processing challenges. It will focus on what is Apache’s Hadoop framework and how it works, also discuss other Apache alternative frameworks, namely Spark and Storm.

APACHE HADOOP

Through time, size of information kept rising and that immense growth generated to change the way this information is processed and managed, as individual processors clock speed evolution slowed, systems evolved to a multi multi-processor or oriented architecture. However there are scenario scenarios, where the data size is too big to be analyzed in acceptable time by a single system, and in this cases where the MapReeduce and a distributed file system are able to shine. Apache Hadoop is a distributed processing infrastructure. It can be used on a single machine, but to take advantage and achieve its full potential, it must , scale it to hundreds or thousands of computers, each with several processor cores processor It's also designed to efficiently distribute large amounts of work and data across multiple systems. It has mainly comprising of two major components, namely Hadoop Distributed File System(HDFS) and MapReduce. HDFS is used to manage data storage, whereas MapReduce is used as programming paradigm.

Hadoop Distributed File System (HDFS)

Hadoop Distributed File System (HDFS) is a system inspired by Google's Google File System [11] stores large files in multiple machines in a shared-nothing mechanism. It is developed to handle big data on clusters of commodity hardware. Apache Hadoop is an open-source implementation of Google's MapReduce [12] and acquired by many large organizations like Google [11], Facebook [13], Yahoo [14], Oracle [15], and Microsoft [16] for enabling their applications on cloud. It provides an abstract and easy programming model to write parallel programs instead of worrying about the data distribution, parallelization, fault tolerance, and computations. MapReduce is a parallel data-processing framework on top of HDFS (provides high throughput and access).Hadoop has the best fault-tolerant, high-throughput, and server-failure survival mechanisms [17].

Like Google File System, Hadoop also maintain replicas of its data splits across different machinesto provide data locality and reliability. Default chunk size of HDFS is 64 MB, and these chunks are once write-multiple-read chunks. Hadoop's master-slave mechanism is shown in Fig 1. It consists of NameNode and DataNodes. A NameNode also called MasterNode, is responsible for controlling the whole MapReduce job through a JobTracker and controlling tasks in a job througha TaskTracker(TT) while working as DataNode (e.g., single node cluster). A DataNode, also called SlaveNode, consists of DataNode and TT. A SecondaryNameNode in large clusters is used to generate snaps of NameNode to avoid loss and works as standby NameNode. NameNode stores meta-data about the files/data chunks stored in DataNodes, while DataNodes store the actual data chunks (64 MB). By default, each data chunk is replicated by a factor of 3.

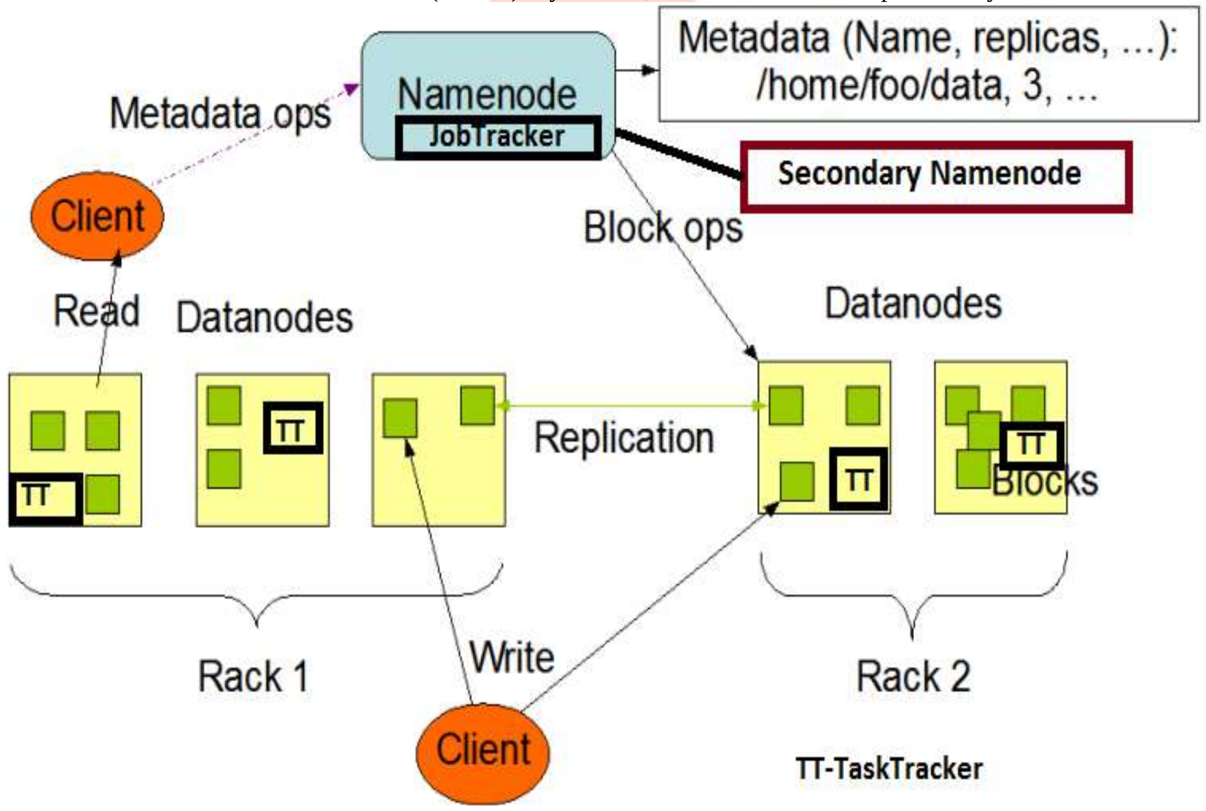

Figure 1: HDFS Architecture

MapReduce

MapReduce is the heart of Apache Hadoop [18]. It is this programming paradigm that allows for massive scalability across hundreds or thousands of servers in a Hadoop cluster. The MapReduce concept is fairly simple to understand for those who are familiar with clustered scale-out data processing solutions. For people new to this topic, it can be somewhat difficult to grasp, because it's not typically something people have been exposed to previously.The term MapReduce actually refers to two separate and distinct tasks that Hadoop programs perform. The first is the map job, which takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). The reduce job takes the output from a map as input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce job is always performed after the map job.

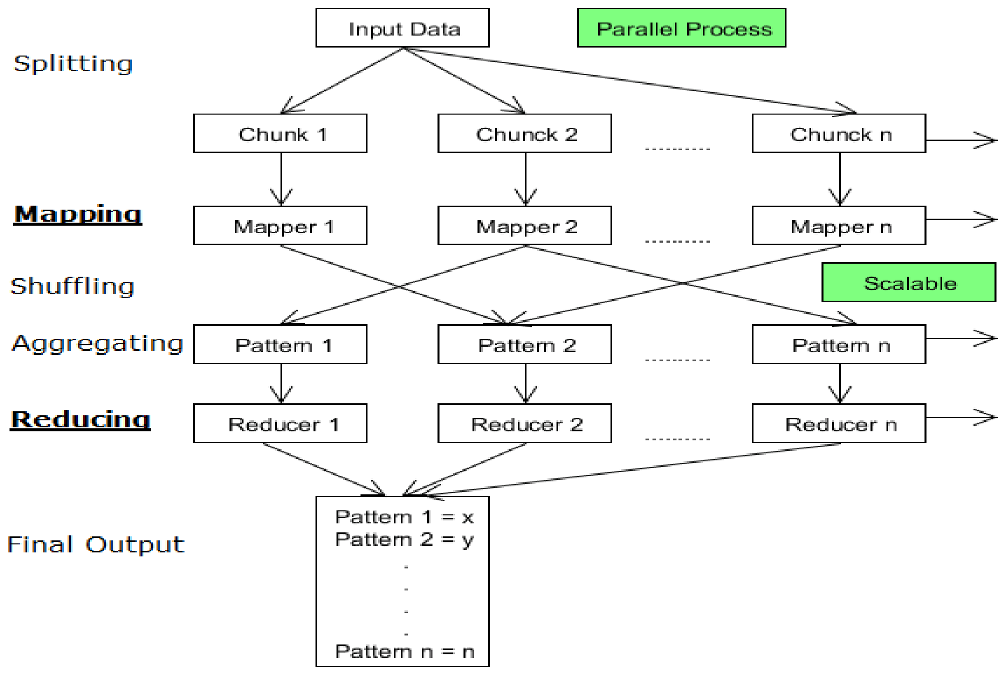

Figure 2: Map Reduce Work Flow

The following steps (Fig 2) summarizes the flow of map reduce algorithm:

1. The input data can be divided into n number of chunks depending upon the amount of data and processing capacity of individual unit.
2. Next, it is passed to the mapper functions. Please note that all the chunks are processed simultaneously at the same time, which embraces the parallel processing of data.
3. After that, shuffling happens which leads to aggregation of similar patterns.
4. Finally, reducers combine them all to get a consolidated output as per the logic.
5. This algorithm embraces scalability as depending on the size of the input data, we can keep increasing the number of the parallel processing units.

APACHE SPARK

Apache Spark[19] is an open source big data processing framework built around speed, ease of use, and sophisticated analytics. It was originally developed in 2009 in UC Berkeley's AMPLab, and open sourced in 2010 as an Apache project.Spark has several advantages compared to other big data and MapReduce technologies like Hadoop and Storm.First of all, Spark gives us a comprehensive, unified framework to manage big data processing requirements with a variety of data sets that are diverse in nature (text data, graph data etc) as well as the source of data (batch v. real-time streaming data).Spark enables applications in Hadoop clusters to run up to 100 times faster in memory and 10 times faster even when running on disk.Spark facilitate user to quickly write applications in Java, Scala, or Python. It comes with a built-in set of over 80 high-level operators. And you can use it interactively to query data within the shell.In addition to Map and Reduce operations, it supports SQL queries, streaming data, machine learning and graph data processing. Developers can use these capabilities stand-alone or combine them to run in a single data pipeline use case.In this first installment of Apache Spark article series, we'll look at what Spark is, how it compares with a typical MapReduce solution and how it provides a complete suite of tools for big data processing.Spark takes MapReduce to the next level with less expensive shuffles in the data processing. With capabilities like in-memory data storage and near real-time processing, the performance can be several times faster than other big data technologies.Spark also supports lazy evaluation of big data queries, which helps with optimization of the steps in data processing workflows. It provides a higher level API to improve developer productivity and a consistent architect model for big data solutions.

Spark holds intermediate results in memory rather than writing them to disk which is very useful especially when you need to work on the same dataset multiple times. It's designed to be an execution engine that works both in-memory and on-disk. Spark operators perform external operations when data does not fit in memory. Spark can be used for processing datasets that larger than the aggregate memory in a cluster. Spark attempts to store as much as data in memory and then will spill to disk. It can store part of a data set in memory and the remaining data on the disk. You have to look at your data and use cases to assess the memory requirements. With this in-memory data storage, Spark comes with performance advantage. There are many Spark features, which include:

- Supports more than just Map and Reduce functions.
- Optimizes arbitrary operator graphs.
- Lazy evaluation of big data queries which helps with the optimization of the overall data processing workflow.
- Provides concise and consistent APIs in Scala, Java and Python.
- Offers interactive shell for Scala and Python. This is not available in Java yet.

Spark is written in Scala Programming Language and runs on Java Virtual Machine (JVM) environment. It currently supports the Scala, Java, Python, Clojure and R .

APACHE STORM

Apache Storm [10] is a distributed real-time big data-processing system. Storm is designed to process vast amount of data in a fault-tolerant and horizontal scalable method. It is a streaming data framework that has the capability of highest ingestion rates. Though Storm is stateless, it manages distributed environment and cluster state via Apache ZooKeeper. It is simple and you can execute all kinds of manipulations on real-time data in parallel.Apache Storm is continuing to be a leader in real-time data analytics. Storm is easy to setup, operate and it guarantees that every message will be processed through the topology at least once.

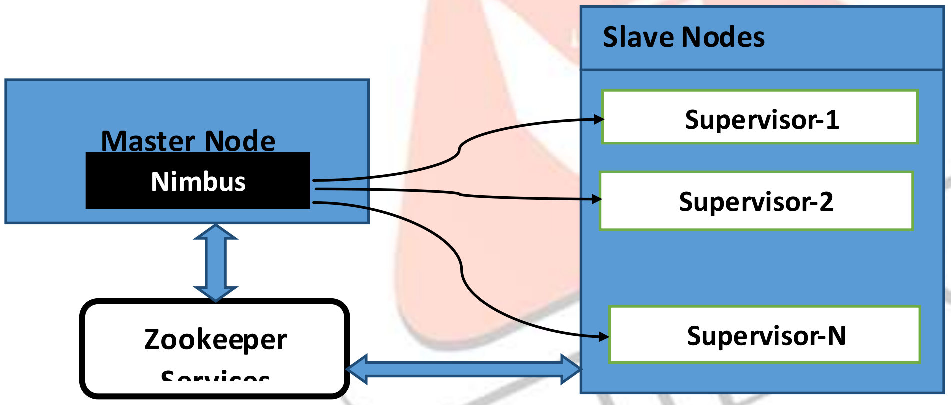

Figure 3: Apache Strom Architecture

There are following two types of nodes services shown in above diagram (Fig. 3)

1. **Nimbus Service on Master Node -** Nimbus is a daemon that runs on the master node of Storm cluster. It is responsible for distributing the code among the worker nodes, assigning input data sets to machines for processing and monitoring for failures. Nimbus service is an Apache Thrift service enabling you to submit the code in any programming language. This way, you can always utilize the language that you are proficient in, without the need of learning a new language to utilize Apache Storm.Nimbus service relies on Apache ZooKeeper service to monitor the message processing tasks as all the worker nodes update their tasks status in Apache ZooKeeper service.
2. **Supervisor Service on Worker Node -** All the workers nodes in Storm cluster run a daemon called Supervisor. Supervisor service receives the work assigned to a machine by Nimbus service. Supervisor manages worker processes to complete the tasks assigned by Nimbus. Each of these worker processes executes a subset of topology that we will talk about next.

APACHE FLINK

Apache Flink[20] is the next generation Big Data tool also known as 4G of Big Data. It is the true stream processing framework (doesn't cut stream into micro-batches). Flink's kernel (core) is a streaming runtime which also provides distributed processing, fault tolerance, etc. Flink processes events at a consistently high speed with low latency; it processes the data at lightning fast speed. It is the large scale data processing framework which can process data generated at very high velocity. Apache Flink is the powerful open source platform which can address following types of requirements efficiently:

- Batch Processing
- Interactive processing
- Real-time stream processing
- Graph Processing
- Iterative Processing
- In-memory processing

Apache Flink[20] is a streaming data flow engine that provides communication, fault-tolerance, and data-distribution for distributed computations over data streams. 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.Apache Flink was started under the project called The Stratosphere. In 2008 Volker Markl formed the idea for Stratosphere and attracted other co-principal Investigators from HU Berlin, TU Berlin, and the Hasso Plattner Institute Potsdam. They jointly worked on a vision and had already put the great efforts on open source deployment and systems building. Later on, several decisive steps had been so that the project can be popular in commercial, research and open source community. A commercial entity named this project as Stratosphere. After applying for Apache incubation in April 2014 Flink name was finalized. Flink is a German word which means swift or agile. The key vision for Apache Flink is to overcome and 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 and out-of-core algorithms, with the MapReduce framework. As Apache Flink is mainly based on the streaming model, Apache Flink iterates data by using streaming architecture. The concept of an iterative algorithm is tightly bounded into Flink query optimizer. Apache Flink's pipelined architecture allows processing the streaming data faster with lower latency than micro-batch architectures (Spark).

III. DISCUSSION

Flink is one of the newest players in the big data computing framework. Flink is an alternative of Map Reduce, it processes data more than 100 times faster than MapReduce. Flink is independent of Hadoop but it can use HDFS to read, write, store, and process the data. Flink does not provide its own data storage system. It takes data from distributed storage. Sparkis another newest players in the MapReduce field. Its purpose is to make data analytics fast to write, and fast to run. Unlike many MapReduce systems (Hadoop inclusive), Spark allows in-memory querying of data (even distributed across machines) rather than using disk I/O. It's no surprise that Spark out-performs Hadoop on many iterative algorithms. Spark is implemented in Scala, a functional object-oriented language that runs on top of the JVM. Similar to other languages like Python and Ruby, Scala has an interactive prompt that users can use to query big data straightfrom the Scala interpreter, making it a good choice in some scenarios. However, it does not support a distributed file system on its own, it depends on Hadoop, if a HDFS is required. The Storm framework is referred as being the Hadoop of Real-time Processing. Hadoop is a batch-processing system, this means, give it a big set of static data and it will do something with it. Storm is real-time, it processes data in parallel as it streams. Therefore, Storm is more a complement to Hadoop rather than a real replacement, as Storm fails when it comes to process large persistent data, as its focus is to be able to process a large number of streams of data (in real time computation), while Hadoop focus is on large amount of persistent data (batch processing).

Framework Features and Summary

This sub-section summarizes the main features and benefits of each of the evaluated frameworks. The similarities and differences of the discussed framework are presented below[21]:

Similarities

The similarities among Hadoop, Spark, Flink and Storm are mentioned below:

- Hadoop, Spark, Flink and Storm are open source processing frameworks.
- Hadoop, Spark, Flink and Stormcan be used for real time BI and big data analytics.
- Hadoop, Spark, Flink and Storm provide fault tolerance and scalability.
- Hadoop, Spark, Flink and Storm are preferred choice of frameworks amongst developers for big data applications (based on the requirements) because of their simple implementation methodology.
- Hadoop, Spark, Flink and Storm are implemented in JVM based programming languages - Java, Scala and Clojure respectively.

Differences

The differences between the four platforms are presented with respected to data processing models, performance and ease of deployment.

Data Processing Models

Hadoop: Hadoop MapReduce is best suited for batch processing. For big data applications that require real time options, organizations must use other open source platform like Impala or Spark. Apache Spark is designed to do more than plain data processing as it can make use of existing machine learning libraries and process graphs. Thanks to the high performance of Apache Spark, it can be used for both batch processing and real time processing. Spark provides an opportunity to use a single platform for everything rather than splitting the tasks on different open source platforms-avoiding the overhead of learning and maintaining different platforms.

Strom: Micro-batching is a special kind of batch processing wherein the batch size is orders smaller. Windowing becomes easy with micro-batching as it offer stateful computation of data. Storm is a complete stream processing engine that supports micro-batching whereas Spark is a batch processing engine that micro-batches but does not render support for streaming in the strictest sense.

Flink: Apache Flink provides a single runtime for the streaming and batch processing

Performance

Spark processes in-memory data whereas Hadoop MapReduce persists back to the disk after a map action or a reduce action thereby Hadoop MapReduce lags behind when compared to Spark in this aspect. Spark requires huge memory just like any other database - as it loads the process into the memory and stores it for caching. However, if Spark runs on top of YARN with various other resources demanding services, then there is a possibility of performance deprivation for Spark. In the case of Hadoop MapReduce, the process is killed as soon as the job is completed, that make it possible to run along with other resource demanding services with just a slight difference in performance.

Similarly, comparing Spark and Storm both provide fault tolerance and scalability but differ in the processing model. Spark streams events in small batches that come in short time window before it processes them whereas Storm processes the events one at a time. Thus, Spark has a latency of few seconds whereas Storm processes an event with just millisecond latency.Spark has good performance on dedicated clusters when the entire data can fit in the memory whereas Hadoop can perform well along other services when data does not fit in memory. Storm is a good option when an application needs sub second latency without data loss whereas Spark can be used in such computations where the event is just processed once.Performance of Apache Flink is excellent as compared to any other data processing system. Apache Flink uses native closed loop iteration operators which make machine learning and graph processing more faster when we compare Hadoop vs Spark vs Flink.

Ease of Development

Hadoop MapReduce is written in Java. Apache Pig makes it easier to develop in Hadoop, although some time needs to be spent on understand and learning the Syntax of Apache Pig. To add the SQL compatibility to Hadoop, developers can use Hive on top of Hadoop. In fact, there are several data integration services and tools that allow developers to run MapReduce jobs without any programming. Hadoop MapReduce lacks the interactive mode but tools like Impala provide a complete package of querying to Hadoop.

Spark uses Scala tuples and they can only be intensified by nesting the generic types because Scala tuples are difficult to be implemented in Java. However, this does not require compromising on the compile time type safety checks. Spark is easier to program as it has interactive mode which is not possible directly with Hadoop. However many tools are coming up to make programming with Hadoop easier. Also, if the project requires an interactive mode for data exploration through API calls - then it is not supported by Storm. Spark has to be used.

Storm uses DAG's which are natural to the processing model. Every node in the directed acyclic graph transforms the data in some way and continues the process. The data transfer between the nodes in directed acyclic graphs has a natural interface and this happens through Storm tuples. However, this can be achieved by compromising at the expense of compile time type safety checks.

Flink iterates data by using its streaming architecture. Flink can be instructed to only process the parts of the data that have actually changed, thus significantly increasing the performance of the job. Apache Flink comes with an optimizer that is independent with the actual programming interface. The Flink optimizer works similarly to a relational Database Optimizer but applies these optimizations to the Flink programs, rather than SQL queries.

IV. CONCLUSION

Hadoop, Spark and Storm have their own benefits, however there are certain aspects like Cost of Development, Performance, and Data Processing models, Message Delivery Guarantees, Latency, Fault Tolerance and Scalability which play a vital role in deciding which one is better for a particular big data application. Hadoop, Spark or Storm can each be a great choice for big data analytics stack and choosing the ideal solution is merely a matter of considering the above mentioned similarities and differences. The beauty of open source tools is that - based on the application requirements, workloads and infrastructure, the ideal choice could be a combination of Spark and Storm together with other open source tools like Apache Hadoop, Apache Kafka, Apache Flume, etc. Based on this research, I understood that the comparison must be made based on use cases oriented view, as the frameworks end up being more complementary than competitive among each other. One thing was made clear, in all references, it does not matter if you choose Hadoop, Spark, Flink or Storm, having the HDFS is an advantage, because it solves many of storage problems associated with big data computing.

Regardless of what open source tools an organization chooses, either it is Hadoop, Spark , Storm, Flink or a combination of either of the four these tools have changed real time business intelligence, as all midsize to large organizations are embracing their advantages.

REFERENCE

[1] Dean J, Ghemawat S.,"MapReduce: simplified data processing on large cluster".,Communications of the ACM,2008, vol. 51, no. 1,pp. 107–113.

[2] Chervenak A, Foster I, Kesselman C, Salisbury C, Tuecke S., "The data grid: towards an architecture for the distributed management and analysis of large scientific datasets", Journal of Network and Computer Applications, 2000, vol. 23, no. 3, pp. 187–200.

[3] Janaki A, KubachT, Loffer M, Schmid U, " Data driven management: bringing more science into management", White Paper, McKinsey Technology Initiative Perspective, 2008, [online] http://www.mckinsey.com/insights/business_technology/big_data_the_next_frontier_for_innovation.

[4]CRA (Computing Research Association), Challenges and opportunities with big data, 2012,[online] http://www.cra.org/ccc/files/ docs/init/bigdatawhitepaper.pdf

[5] Advancing discovery in science and engineering, the role of basic computing research, 2013,[online] http://www.cra.org/ccc/files/ docs/Natl_Priorities/web_data_spring.pdf

[6] IDC, The digital universe in 2020: big data, bigger digital shadows, and biggest growth in the Far East,2014,[online] www.emc. com/leadership/digital-universe/index.htm

[7]Survey of big data architectures and framework from the industry, NIST big data public working group,2014,[online] http:// jtc1bigdatasg.nist.gov/_workshop2/07_NBD-PWD_Big_Data_Architectures_Survey.pdf

[8]. Mayer VV, Cukier K., Big Data: A Revolution That Will Transform How We Live, Work and Think. John Murray Ress: UK, 2013.

[9] Ginsberg J.,“Detecting influenza epidemics using search engine query data”. Nature , 2009, vol.457, pp. 1012–1014.
[10] Yahoo! Hadoop Tutorial, 2014 [online] https://developer.yahoo.co com/hadoop/tutorial/. Accessed 20 m/Dec 2014.
[11] Ghemawat S, Gobioff H, Leung ST.,“The Google file system”,ACM SIGOPS Operating Systems Review,2003, vol.37, no.5,pp.29–43.
[12] Dean J, Ghemawat S., “MapReduce: simplified data processing on large clusters”, Communications of the ACM , 2008 vol.51, no. 1, pp.107–113.
[13] Borthakur D.,“Facebook has the worlds largest Hadoop cluster” [online]. http://hadoopblog.blogspot.in/2010/05/facebook-has-worlds-largest-hadoop.html
[14] Kaushik RT, Bhandarkar M.,“GreenHDFS: towards an energy-conserving, storage-efficient, hybrid Hadoop compute cluster”. Proceedings of the USENIX Annual Technical Conference, Boston, MA, 2010.
[15] Dijcks JP.,“Oracle: big data for the enterprise”,Oracle White Paper, 2012.
[16] Gunarathne T, Wu T-L, Qiu J, Fox G., “MapReduce in the clouds for science”, In 2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom). IEEE: Bloomington, IN, USA, 2010.
[17] Jalote P, Jalote P. Fault Tolerance in Distributed Systems. PTR Prentice Hall: Englewood Cliffs, 1994.
[18] Apache MapReduce,2015[online] http://hadoop.apache.org/docs/stable/mapred_tutorial.html.
[19] Apache Spark, 2014,[online] https://spark.apache.org/. Accessed 26 Dec 2014.
[20] Apache Flink, 2015,[online]https://flink.apache.org/
[21] Spark vs Hadoop vs Storm,2014,[online] https://www.dezyre.com/article/spark-vs-hadoop-vs-storm/145.