

## A COMPREHENSIVE SURVEY OF BIG DATA TECHNOLOGY TOOLS

1.Nana kwame Gyamfi/University of Ghana: dept. of computer Science: Accra-Legon, Ghana

3.Dr. Jamal-Deen Abdllai/University of Ghana: dept. of computer Science: Accra-Legon, Ghana

2.Dr. Ferdinand Katsriku/University of Ghana: dept. of computer Science: Accra-Legon, Ghana

**Abstract**— As the world’s data is growing rapidly and traditional tools for machine learning are coming insufficient as we moved toward distributed and real-time processing. The task of selecting machine learning tools for big data has become difficult. The available tools have drawbacks and advantages, and others have imbrications uses. The main goal of this paper is to aid the researcher and professional who understand machine learning but is inexperienced with big data. In order to access tools, one should have broad understanding of what to look for. On the grounds of that, this paper provides a list of criteria for making selection along with comprehensive analysis of the advantage and drawbacks of each. We focus on discussing the advantage and disadvantage of different processing paradigms along with a comparison of engines that implement them, including MapReduce, Spark, Flint and Storm. No toolkit is 100% efficiency, so this paper aims to help make decisions confine by providing as much information as possible.

**Keywords**-Big data; Hadoop; Spark; Storm; MapReduce; Machine Learning

### I. INTRODUCTION (MACHINE LEARNING FLOWWORK)

The term ‘data tsunami’ is often used to describe the overwhelming volume of data that is currently being experienced in the business environment. Such a deluge is now beyond the capabilities of traditional relational databases. New tools to filter, classifiers, read the content of hiding information will be required. According to Katal Avita [7], the technical interpretation of big data in Data Engineering /I.T operational level means “ the growth of data in a magnitude and at a speed which makes difficult to to be manage my any Computer Memory, Processor(the processing power) and computer architecture”.

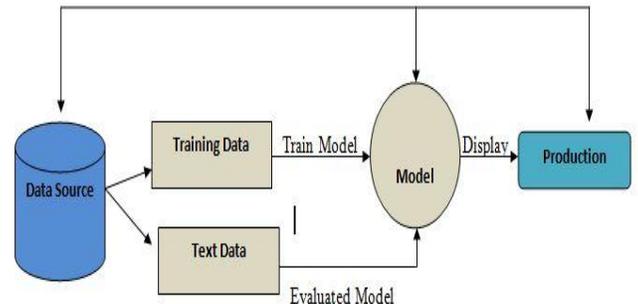


Fig.1 Machine Learning Workflow

### FIG.1 MACHINE LEARNING WORKFLOW

Big Data is the expression used to describe massive volumes of structured and unstructured data that are so large that it is very difficult to process this data using traditional databases and software technologies. In 2012, Gartner retrieved and gave a more detailed definition as: "Big Data are high volume, high-volume and high-variety information assets that require new forms of processing to enable enhanced decision making, insight discovery and process optimization". The term "Big Data" is believed to have originated from web searching companies who had to query loosely structured very large distributed data. Demchenko, Laat and Membray[10], based on the Gartner definition give and improved definition of Big Data as "technologies that are targeting to process high-volume, high-velocity data (sets/assets) to extract intended data value and ensure high veracity of original data and information processing for enhance insight, decision making and process control; all those demands new infrastructure services and tools that allow obtaining data from variety of sources (including sensor networks) and delivering data in a variety of forms to different data information consumers and devices.

As theoretical research is leverage into practical tasks, machining learning tools are rapidly increasingly seen as not just useful, but integral to many business operations, coined by

Sara et al, [9]. Machine learning is purposely to help systems to learn from the past or present and use the knowledge acquired to make decision or predictions regarding unknown future events. Figure 1, above shows machine learning workflow. Big data is bringing machine learning to forefront of research and industry applications. According to International Data Corporation's annual Digital Universe study [1], the amount of data on our planet is set to reach 44 zettabytes ( $4.4 * 10^{22}$  bytes) by 2020 which would be ten times larger it was in 2012. Many industries are still generating large amount of data to processed efficiently using traditional techniques. With such kind of growth data production, challenge faced by the machine learning community is how to best efficiently process and learn from big data. Tools like Weka, R and Orange, which are the most popular machine learning toolkits can't effectively process this data due to the workload. Weka and Orange have distributed implementation of some machine learning algorithms available; they are not on the same level as tools that were initially designed and built for terabyte-scale. Hadoop, a popular framework for working with big data, helps to solve this scalability problem by offering distributed storage and processing solutions. Hadoop framework, has provide extensible platform that allows for many machine learning projects and applications, the main aim of this paper is to present those tools.

## 1. Problem Statement

We live in an age where data is growing orders of magnitude faster than ever before, there is need for organizations and business entities to find out the right choice of toolkits for analyzing and processing their data. With the most algorithms implemented with these toolkits for big data, make it difficult for organizations and enterprise to make right choice.

The heart of machine learning is the data that powers the model, but data generated from these organizations differ. Therefore, there is need for organizations to know the kind of tool kit that will appropriately address solution to their big data.

There is a lack of comprehensive research on many of these big data tools, despite being widely use. The goal of this paper is to facilitate these decisions by providing a comprehensive review of these current open source scalable tools for machine learning.

## 2. Big Data

Te term "big data" has become a buzzword and as such, it is often overused and misunderstood. While the frameworks we discuss in this paper are able to effectively process data of varying sizes and complexities, they were designed with very large data in mind and may not be the best choice for certain smaller projects.

There is no universal agreed-upon definition of big data. Big Data is the expression used to describe massive volumes of structured and unstructured data that are so large that it is very difficult to process this data using traditional databases and software technologies. In 2012, Gartner retrieved and gave a more detailed definition as:"Big Data are high volume, high-volume and high-variety information assets that require new forms of processing to enable enhanced decision making, insight discovery and process optimization".

Demchenko et al [10], based on the Gartner definition give and improved definition of Big Data as "technologies that are targeting to process high-volume, high-velocity data (sets/assets) to extract intended data value and ensure high veracity of original data and information processing for enhance insight, decision making and process control; all those demands new infrastructure services and tools that allow obtaining data from variety of sources (including sensor networks) and delivering data in a variety of forms to different data information consumers and devices." This modified definition by Demchenko et al, [10] defines some basic characteristics of Big Data which is termed as "Vs" of Big Data. Pandey and Tokekar [8], looks at the basic characteristics of Big Data as "3V".

Traditionally big data has been defined in terms of the three V's, Volume, Velocity and Variety with the following properties.

- a. Volume: Volume is the sum of all the data generated from all different sources per unit time. Many factors contribute towards increasing volume- storing data, live streaming data and data collected from various sources and sensors etc.
- b. Variety: Variety is provided for through the many different data sources each generating some amount of data, whose form and structure may be different from all other, per unit time. In our recent world data comes in all types of formats- from traditional databases, text documents, emails, videos, audio, transactions etc.,
- c. Velocity: This means the speed with which new data is generated and how quickly the data is process in other words how fast the data is being produced and how fast the data to be processed to meet the demand.

In the years since Gartner's papers was published, numerous people have proposed additions to the list and many refer to four or five V's, adding in Value or Veracity [10].

Distributed storage system has come to solve the problem of big data collection, which is well designed to carefully control and management in a fault-tolerant manner. Parallelization of algorithms is another solution to big data objects, which accomplished in one or two ways data parallelism. In which

the data is divided into more manageable pieces and each subset is computed simultaneously, in which algorithm is divided into steps that can be performed concurrently.

## 2.1 Hadoop Ecosystem

Hadoop, which is a free, Java-based programming framework, supports the processing of large sets of data in a distributed computing environment. It is a part of the Apache project sponsored by the Apache Software Foundation. Hadoop cluster uses a Master/Slave structure (Huang & Ting-tin et al). Using Hadoop, large data sets can be processed across a cluster of servers and applications can be run on systems with thousands of nodes involving thousands of terabytes. Distributed file system in Hadoop helps in rapid data transfer rates and allows the system to continue its normal operation even in the case of some node failures.

The Hadoop platform consists of three key services: a reliable, distributed file system called Hadoop Distributed File System (HDFS), the high-performance parallel data processing engines like Spark, Storm and Hadoop MapReduce. Hadoop was created by Doug Cutting and named after his son's toy elephant. Vendors that provide Hadoop-based platforms include Cloudera, Hortonworks, MapR, Greenplum, IBM, and Amazon.

The combination of HDFS and MapReduce provides a software framework for processing vast amounts of data in parallel on large clusters of commodity hardware (potentially scaling to thousands of nodes) in a reliable, fault-tolerant manner. Hadoop is a generic processing framework designed to execute queries and other batch read operations against massive datasets that can scale from tens of terabytes to petabytes in size.

The popularity of Hadoop has grown in the last few years, because it meets the needs of many organizations for flexible data analysis capabilities with an unmatched price-performance curve. The flexible data analysis features apply to data in a variety of formats, from unstructured data, such as raw text, to semi-structured data, such as logs, to structured data with a fixed schema.

Hadoop has been particularly useful in environments where massive server farms are used to collect data from a variety of sources. Hadoop is able to process parallel queries as big, background batch jobs on the same server farm. This saves the user from having to acquire additional hardware for a traditional database system to process the data (assume such a system can scale to the required size). Hadoop also reduces the effort and time required to load data into another system; you can process it directly within Hadoop. This overhead becomes impractical in very large data sets.

The general structure of the ecosystem can be described in terms of three layers: storage, processing, and management.

While the primary focus of this paper is on tools that reside in the processing layer, it is important to understand the context of how they can be used in a workflow by looking at the makeup of the ecosystem as a whole.

- The storage layer: it resides at the lowest level of this stack, and by default includes the HDFS. There are also a variety of other options for distributed data storage which run on top of HDFS or work as standalone systems. HDFS is known for its scalability and fault tolerance, and good option for historical data does not need to be edited or accessed frequently, not only is also good for fast random reads or writes.
- Processing layer: processing layer is where the actual analysis takes place. The foundation of this layer is YARN, which allows one or more processing engines to run on a Hadoop cluster. Processing engines will be discussed in detail in the next section. In addition to the processing engines, this layer includes a number of different tools that can be used for machine learning and data analysis
- The management layer: The management layer includes tools for user interaction and high-level organization. These include scheduling, monitoring, coordination, and user interface. Oozie is a workflows scheduler, manages jobs for many of the tools in the processing layer, including processing engines, Pig, Sqoop, and Hive, among others. For complex workflows which require multiple jobs and tools, it specifies a sequence of actions and coordinates between them to complete the tasks. It also facilitates scheduling of jobs which need to run on regular intervals.

## 3. Data Processing Engines

In more recent years, MapReduce has begun to fall out of favor, particularly in the machine learning community, due to its high overhead costs, lack of speed, and the fact that many machine learning tasks do not easily fit into the MapReduce paradigm. In 2014, Google announced that it was being phased out in favor of other projects [2]. Since MapReduce has been decoupled from Hadoop through YARN (Yet Another Resource Negotiator), it is now a lot

easier to work with a new engine on an existing cluster, and over the course of the past few years, a number of projects have been introduced that attempt to solve the issues inherent in MapReduce.

The processing models used for many of these may be categorized as either batch or streaming. A third model, known as bulk-synchronous parallel (BSP), is used for iterative graphing tasks, but will not be discussed in detail in this paper. While graph algorithms are related to ML, they are used more for traditional analytics and the focus of this paper is on other types of learning tasks. Examples of tools which employ the BSP model are Apache Giraph [4] and Apache Hama [5].

The remainder of this section will discuss some of the more widely used projects which leverage the batch and streaming paradigms. A high-level overview of these projects is in Table 1. In addition to the underlying processing approach used, here are several important considerations for evaluation of these tools:

- **Latency:** This refers to the amount of time between starting a job and getting initial results. Speed may not be important for every project. If a project is not time-sensitive, a batch system may be preferred for its simplicity, but for projects that require real time or near-real time results, a streaming platform would be advised.
- **Throughput:** Throughput measures the amount of work done over a given time period. This can be thought of as a measure of efficiency.
- **Fault tolerance:** All of the platforms discussed in this paper are fault-tolerant but the methods which they use to achieve that may vary. We look at the mechanisms that are in place to detect failures, as well as how the platform is able to recover after such a failure occurs.
- **Usability:** Despite the interfaces and abstractions discussed in the previous section and libraries for machine learning that will be discussed later in this paper, the typical user will spend a good deal of time interacting with the engine itself. With this in mind we ask, how difficult is it to install and configure? What interface language(s) does it use? How difficult is it to program for?
- **Scalability:** All of the processing engines discussed in this paper were designed to be scalable, but the different methods employed have varying degrees of success. For this reason, it is important to examine whether there are bottlenecks when data input or cluster sizes grow. Additionally, these engines

were designed for very large data, but many real-world use cases involve at least some processing of smaller datasets. We look at how these are handled as well.

	Current stable release (as June 1, 2016)	Execution model	Supported languages	Associated ML tools	In-memory processing	Low latency	Fault tolerance	Enterprise support
MapReduce	2.27	Batch	Java	Mahout	*	*	√	*
Spark	2.0.0	Batch Streaming	Java, Python, Scala, R	MLlib, Mahout, H2O	√	√	√	√

Table 1. Data Processing engines for Hadoop

### 3.1 MapReduce

MapReduce provides the interface for distribution of sub-task and gathering of outputs. Is the heart of Hadoop, it is a 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. In the map phase, [9] explains that, input data is split into independent sub-programs and distributes it from the master node to the workers node for parallel processing in smaller units and back to the master node. Its approach to machine learning perform batch learning, in which the training data set is a lack of efficiency in terms of speed and computational resources. The trained data is read from HDFS. It is

compatible with the Mahout library for ML, and programming language used is Java. The fault tolerance mechanism employed by MapReduce is achieved through data replication, which can affect scalability by increasing the size of data even further. Again MapReduce does not easily allow for iterative processing, making it unsuitable for many machine learning projects. It does not support in-memory processing of data.

### 3.2 Spark

A fast and general compute engine for Hadoop data. Spark provides a simple and expressive programming model that supports a wide range of applications, including ETL, machine learning, stream processing, and graph computation.

Spark is the heir apparent to the Big Data processing kingdom. Spark and Hadoop are often contrasted as an "either/or" choice, but that isn't really the case. The Hadoop ecosystem can accommodate the Spark processing engine in place of MapReduce, leading to all sorts of different environment make-ups that may include a mix of tools and technologies from both ecosystems. As one specific example of this interplay, Big Data powerhouse Cloudera is now replacing MapReduce with Spark as the default processing engine in all of its Hadoop implementations moving forward. As another example, Spark does not include its own distributed storage layer, and as such it may take advantage of Hadoop's distributed filesystem (HDFS), among other technologies unrelated to Hadoop (such as Mesos) [11].

Spark differs from Hadoop and the MapReduce paradigm in that it works in-memory, speeding up processing times. Spark also circumvents the imposed linear dataflow of Hadoop's default MapReduce engine, allowing for a more flexible pipeline construction [11].

### 3.3 Storm

Apache Storm is a distributed real-time computation system, whose applications are designed as directed acyclic graphs. Storm is designed for easily processing unbounded streams, and can be used with any programming language [12]. It has been benchmarked at processing over one million tuples per second per node, is highly scalable, and provides processing job guarantees. Unique for items on this list, Storm is written in Clojure, the Lisp-like functional-first programming language.

Apache Storm can be used for real-time analytics, distributed machine learning, and numerous other cases, especially those of high data velocity. Storm can run on YARN and integrate into Hadoop ecosystems, providing existing implementations a solution for real-time stream processing. Five characteristics which make Storm ideal

for real-time processing workloads are (taken from HortonWorks):

- Fast - benchmarked as processing one million 100 byte messages per second per node
- Scalable - with parallel calculations that run across a cluster of machines
- Fault-tolerant - when workers die, Storm will automatically restart them. If a node dies, the worker will be restarted on another node.
- Reliable - Storm guarantees that each unit of data (tuple) will be processed at least once or exactly once. Messages are only replayed when there are failures.
- Easy to operate - standard configurations are suitable for production on day one. Once deployed, Storm is easy to operate.

Keep in mind that Storm is a stream processing engine without batch support. Storm does not support state management natively; however, Trident, a high level abstraction layer for Storm, can be used to accomplish state persistence. Trident also brings functionality similar to Spark, as it operates on mini-batches.

### 3.4 Flink

Apache Flink is a streaming dataflow engine, aiming to provide facilities for distributed computation over streams of data. Treating batch processes as a special case of streaming data, Flink is effectively both a batch and real-time processing framework, but one which clearly puts streaming first [13].

Flink provides a number of APIs, including a streaming API for Java and Scala, a static data API for Java, Scala, and Python, and an SQL-like query API for embedding in Java and Scala code. It also has its own machine learning and graph processing libraries. Flink has an impressive set of additional features, including:

- High Performance & Low Latency
- Support for Event Time and Out-of-Order Events
- Exactly-once Semantics for Stateful Computations
- Continuous Streaming Model with Backpressure
- Fault-tolerance via Lightweight Distributed Snapshots

Flink is truly stream-oriented. Spark operates in batch mode, and even though it is able to cut the batch operating times down to very frequently occurring, it cannot operate on rows as Flink can. If you are processing stream data in real-time (real real-time), Spark probably won't cut it.

## 5.0 Conclusion

Current trends in technology, such as Internet of Things and increase adoption of wearable computers are allowing for unprecedented access to massive amounts of heterogeneous data. Efficient learning from this data requires complex

architectures that utilize combination of tools and techniques for collection, storage processing and analysis [6]. This has increasingly adopted research in machine learning platform to come out with tools with efficient algorithms.

We chose to focus the bulk of our paper on processing engines of the hadoop ecosystem for two reasons. First, they are among the most innovative we have seen. Lastly, an Apache project tends to draw large number of active users who are

REFERENCES

[1]. International Data Corporation. Digital Universe Study. 2014. <http://www.emc.com/leadership/digital-universe/index.htm>. Accessed 1 Jun 2015

[2] DeMichillie G. Reimagining developer productivity and data analytics in the cloud—news from Google IO. 2014. <http://googlecloudplatform.blogspot.com/2014/06/reimagining-developer-productivity-and-data-analytics-inthe-cloud-news-from-google-io.html>. Accessed 5 Jan 2015.

[3] Demchenko Y, Grosso P, de Laat C, Membrey P. Addressing big data issues in scientific data infrastructure. In: 2013 International Conference on Collaboration Technologies and Systems (CTS), San Diego, 2013. IEEE, pp 48–55

[4] Apache Giraph. <http://giraph.apache.org/>

[5] Apache Hama. <https://hama.apache.org/>

[6] Ganz F, Puschmann D, Barnaghi P, Carrez F. A practical evaluation of information processing and abstraction techniques for the internet of things. IEEE Internet Things J; 2015

[7] A Katal, Wazid M and R.H Goudar, "Big data: Issues, challenges, tools and Good practices", *Noida 2013*, pp. 404-409, 2013

[8] Pandey, S., & Tokekar, V. (2014). Prominence of MapReduce in Big Data Processing. *IEEE Computer Society*, 555-560.

[9] Sara, d. R., Lopez, V., Jose, M., & Herrera, F. (2014). On the use of MapReduce for imbalanced big data using Random forest. *Elsevier - Information Science*, 112-137.

[10] Demchenko, Y., Laat, C. d., & Membrey, P. (2014). Defining Architecture Component of Big Data Ecosystem. *IEEE*, 104-112.

[11] Apache Spark. <http://spark.apache.org/>

[12] Apache Storm. <http://storm.apache.org/>

[13] Apache Flink. <http://flink.apache.org/>

AUTHORS PROFILE

Strong academic background in Computer Science, Cyber Security and Information Technology, combined with skills in Data Science graduate studies, given me edge to participate in any Data Engineering and Institutional Data Integration project and training; and the ability of leveraging technical and business solutions for the facilitation of enterprise-wide strategies.

[ngyamfi224@gmail.com](mailto:ngyamfi224@gmail.com)

+233207443714

MapReduce	2.27	Batch	Java	Mahout	*	*	√	*
Spark	2.0.0	Batch Streaming	Java, Python, Scala, R	MLlib, Mahout, H <sub>2</sub> O	√	√	√	√
Storm	1.0.2	Streaming	Any	SAMOA	√	√	√	*
Flink	0.8.4	batch	Java Scala	SAMOA, Flink-ML	√	√	√	*