© Universiti Tun Hussein Onn Malaysia Publisher's Office

IJIE

Journal homepage: http://penerbit.uthm.edu.my/ojs/index.php/ijie

The International Journal of Integrated Engineering

A Survey on Vertical and Horizontal Scaling Platforms for Big Data Analytics

Ahmed Hussein Ali[1,*], Mahmood Zaki Abdullah[2]

[1]Ph.D. candidate,
ICCI, Informatics Institute for Postgraduate Studies, Baghdad, IRAQ

[2]Department of Computer Engineering,
Al-Mustansiriyah University, Baghdad, IRAQ

*Corresponding Author

DOI: https://doi.org/10.30880/ijie.2019.11.06.015
Received 22 June 2018; Accepted 8 October 2018; Available online 12 September 2019

Abstract: There is no doubt that we are entering the era of big data. The challenge is on how to store, search, and analyze the huge amount of data that is being generated per second. One of the main obstacles to the big data researchers is how to find the appropriate big data analysis platform. The basic aim of this work is to present a complete investigation of all the available platforms for big data analysis in terms of vertical and horizontal scaling, and its compatible framework and applications in detail. Finally, this article will outline some research trends and other open issues in big data analytics

Keywords: Big data, Scalability, H2O, Graphics processing units, Big data frameworks, Big data platforms.

1. Introduction

For many years, the big data phenomenon was surprising neglected by researchers, but today, with the increasing amount of data to the tone of up to 35 trillion GB by 2020, it is mandatory to focus on this phenomenon. Clearly, we are living in the big data era [1, 2]; big data is not just about size, it is also about the complexity, noisy, heterogeneity, streaming, and verity of data. The large scale of data opened the domain for many studies in social networks, commerce, science, engineering, and security. The roadmap of this article is presented in Fig. 1.

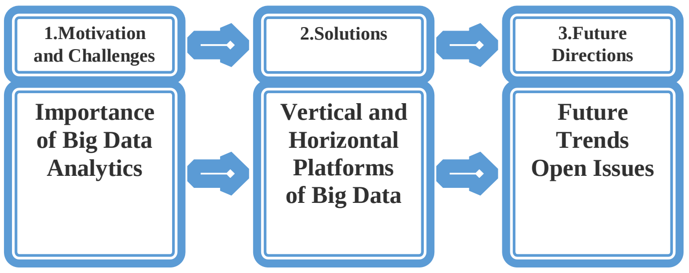
1.Motivation and Challenges
2.Solutions
3.Future Directions
Importance of Big Data Analytics
Vertical and Horizontal Platforms of Big Data
Future Trends Open Issues

Fig. 1 - Roadmap of this paper.

2. Understanding Big Data

Scientific research has been revolutionized by big data. Indirectly, big data has become the most significant source of knowledge that helps in making important decisions. In this section, the challenges of big data will be addressed through three important questions; firstly, what is big data? secondly, what are the properties of big data? and thirdly, what are the characteristics of big data? The term 'big data' refers to a huge amount of dataset that cannot be processed using the traditional database software tools. These data are sourced from the public web, social media, docs, videos, audios, pictures, sensor data, machine log data, and archives. Big data is often used to describe all types of data captured over time. About 2.5 Exabyte ($2.5 * 10^{18}$) of data have been reportedly generated over time [3]. The properties of big data come with three concepts: structured, semi-structured, and unstructured data [4]. Briefly, the structured data refers to all data that can be stored in the SQL database. They are arranged in rows and columns and have relational keys. Sometimes, they are referred to as fully structured data (Fig. 2). Structured data represent only 5 to 10 % of big data. Semi-structured data (Fig. 3) is often referred to as scheme less or self-describing data, or data that cannot be set up in relational databases such as XML, HTML, and RDF data. They represent only 5 to 10% of big data as well.

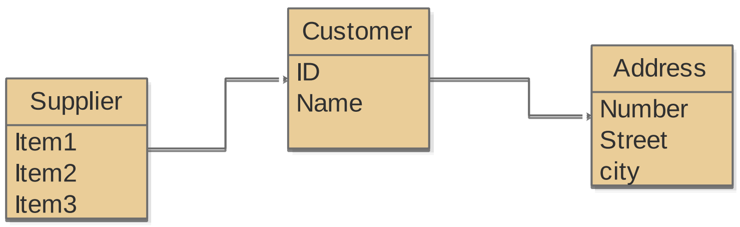

Table: Customers

ID	Name	Age

Fig. 2 - Structured data.

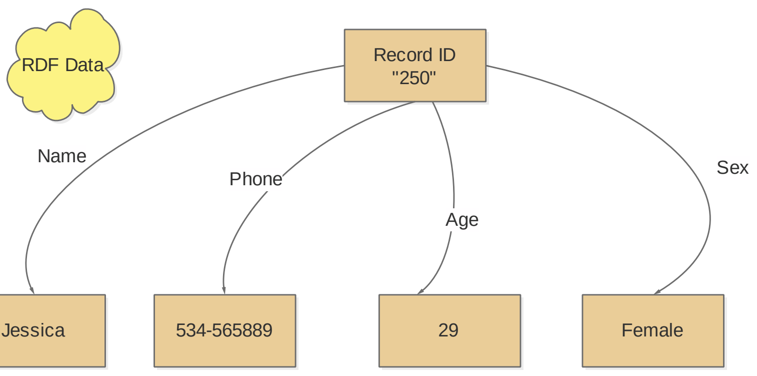

Fig. 3 - Semi-structured data (RDF data).

Unstructured data lacks any identifiable structure, and their storage in the relational databases cannot be done in rows and columns. They are usually in the form of texts, audio files, videos, photos, e-mail messages, web pages, and presentations. Unstructured data (Figure 4) represents around 80 % of big data [5].

Fig. 4 - Unstructured data.

For decades, databases have been the central hub of data. Data is processed and manipulated through CRUD (Creat, Read, Update, Delete) operations that are expressed by the Structured Query Language (SQL) of databases. However, compared to the newly emerged systems earlier mentioned, traditional databases fall short of complex analytics due to these limitations:

1. Data representation mismatch: It is extremely hard to express complex analytics computation in pure SQL. Databases are designed to work with tables with several rows but a limited number of attributes/columns. Data analytics, however, works with vectors and matrices (dense or sparse) that potentially contain millions, if not billions, of dimensions. Thus, vectors and matrices must be stored in a “relational” way which is unintuitive and hard to work with.
2. Lack of numerical operations: Several basic operations, such as dot-product and matrix multiplication which is fundamental to complex analytics are not supported in SQL. A possible remedy is to implement these operations in a series of join-group-by aggregates assuming the vectors are represented as tables, though such solution suffers from suboptimal performance.
3. Missing iterative support: Complex analytics typically requires going over the data multiple iterations and memorizing the states between iterations. However, there is no SQL semantics of “iteration” and transferring states between consecutive queries.

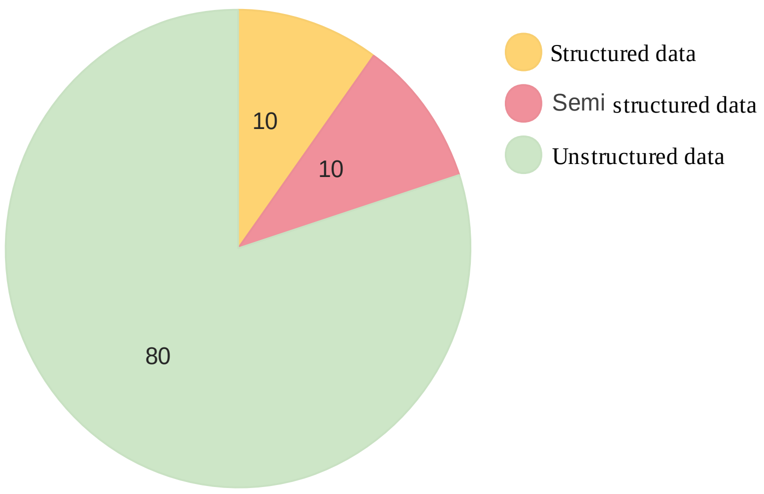

Fig. 5 - Pie of big data percentages.

4. No control in data access order: Complex analytics often requires random access to data, whereas SQL does not specify the order of data to be processed.
5. Other limitation: Numerical solvers of complex analytics needs to specify certain hyperparameters, such as the step-size to run properly. Hyper-parameter tuning for complex analytics solvers remains a human-in-the-loop process that is costly.

One of the facts that all the proposed definitions of big data in the literature agree with is the six V’s (volume, velocity, veracity, variety, value, variability) of big data. Practically, there are eight V’s which are divided into:

- Generic big data characteristics -

- Volume (large scale, terabytes, distributed).
- Variety (nonlinear, heterogeneous, multifactor, dynamic, high dimension).
- Velocity (real/near-time, high speed, streams).

▪ Acquired characteristics after entering a system-
- Value (correlation, low-value density, diverse data meaning).
- Veracity (trustworthiness, incomplete, uncertain).
- Variability (linkage, change data, change model).
- Virality describes how quickly information gets spread across a network. The rate of speed is measured with respect to time.
- Viscosity measures the confrontation to flow in the volume of data.

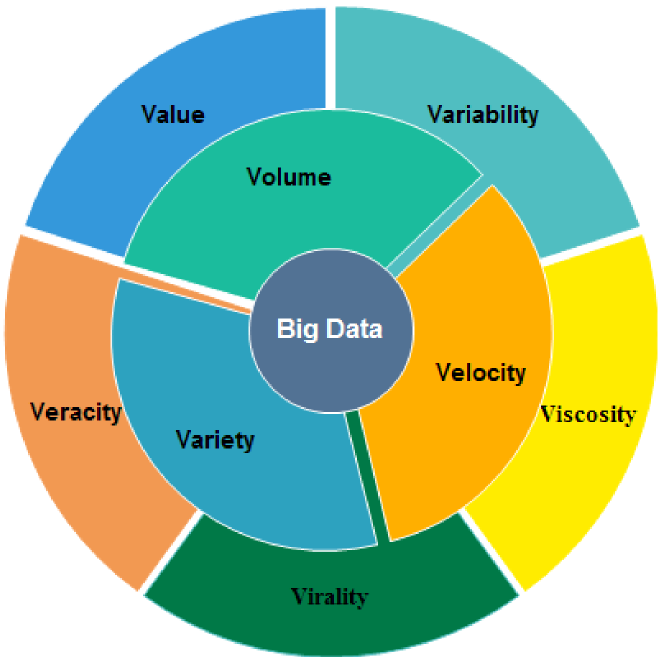

Fig. 6 - Big data challenges.

3. Similarities Between Different Big Data Analytics Platforms

Generally, all the hardware and software tools required to run an application are provided by the platform. A collection of different libraries/classes that have a common goal and perform a common task is referred to as a framework. Several big data frameworks with different features exist; the selection of the proper platform demands a complete information of the capabilities of the existing frameworks, especially their adaptability to increased demands in data processing which play a significant role in selecting the right framework to build the analytics based on the solutions available on that platform [6]. For big data analytics, various solutions have been proposed, and these solutions can be divided [7] into (1) Processing/Compute: Hadoop [8], Nvidia CUDA [9], or Twitter Storm [10], (2) Storage: Titan or HDFS, and (3) Analytics: MLPACK [11] or Mahout [12]. Even though there are several commercial data analysis products, tools and platforms presented by recognized organizations can still be easily found. The large demand for computing power and storage is easily met using cloud computing technologies on these platforms and frameworks [13]. According to [14], the cloud-based programs for data analysis can be grouped into data analysis platform (i) as a service, (ii) as a service, and (iii) infrastructure as a service. Furthermore, [15] proposed a generalized big data analytics architecture which comprised of big data collection from different sources, intra/inter big data processing, and distributed big data storage. Being that several data analytics platforms and frameworks have been proposed, attempts have been made to compare these platforms to provide a guide when selecting applicable platforms or frameworks for relevant works. When introducing big data analytics in terms of platform, Cuzzocrea et al. [16] first discussed the response of the recent studies to the computational emergency issues of big data analysis. Several unattended issues like uncorrelated data filtering and data source heterogeneity and possible research directions were similarly addressed in the study. Zhang and Huang [17] explained the kind of method and framework needed to different big data approaches using 5Ws model. They further stated that the 5Ws model represents the type of data, the reason for their usage, the source of the data, when the data occur, the recipient of the data, and how the data will be transmitted. Later, [18] compared Hadoop, Drill, and Storm [19] using workload, owner, low latency, source code, and complexity as features. From the comparison, the Apache Hadoop framework was evidently better in latency performance compared to the other two frameworks. Chalmers et al. provided a better understanding of the advantages and disadvantages of big data solutions by employing variability, volume, skill, velocity, infrastructure, and user skill to evaluate eight big data analytics solutions.

4. Scaling

System scaling refers to the capability of a system to tolerate an increase in demands for data processing. Different forms of scaling platforms for big data processing can be grouped as follows:
•Horizontal Scaling: This involves workload distribution over several servers (these may even be commodity machines). This process is also referred to as “scaling out”. It involves the piling of multiple independent machines to enhance their data processing power. Several operating systems are typically run on separate machines.
•Vertical Scaling: This involves the installation of additional processors, faster hardware and memory into a single server. The process is also known as “scaling up” and it often requires one operating system.

4.1 Horizontal Scaling Platforms

Some of the known platforms for horizontal scaling include Apache Hadoop and Map Reduce. Researchers are recently working on the development of tools for next generational horizontal scaling, such as Spark [20]. These tools will overcome the drawbacks of the existing platforms. Further discussions will now be provided for each of these platforms in this section.

Hadoop

Hadoop was specifically designed for HDFS and MapReduce concepts; both are commonly associated with distributed computation. It is known that MapReduce is the core of Hadoop which carry can parallelly process distributed data. HDFS is a UNIX-based data storage layer of Hadoop and is seen as Hadoop's own rack-aware filesystem. HDFS is based on the Google filesystem concept. One major attribute of Hadoop is its ability to partition computational processes cross several hosts, as well as its ability to execute parallel computations on applications close to their data. Data files on HDFS are duplicated as block sequences in the cluster. A Hadoop cluster simply adds up commodity servers to scale up storage capacity, I/O bandwidth, and computation capacity. There are different ways HDFS can be accessed from applications as they ordinarily provide Java API for use in applications. The Yahoo! Hadoop clusters span approximately 40,000 servers and can store about 40 petabytes of application data. The largest Hadoop cluster is 4,000 servers. Also, about 100 other organizations globally are known to use Hadoop.

Some of the characteristics of HDFS include the ability to tolerate faults, can run on commodity hardware, can handle large datasets, can master-slave paradigms, and ability to write files with one access only. The advantages and disadvantages of Hadoop include:

Advantages:

1) The range of data sources

Data sourced from multiple sources are either structured or unstructured. These sources can be emailed conversations, social media, or clickstream data. Much time would be required to convert the multiple sourced data into a single format. With Hadoop, this conversion is not needed as it can derive valuable data from any type of data. It can also perform other functions like fraud detection, data warehousing, and market campaign analysis.

2) Cost-effective

Companies using the conventional methods spend much of their benefits in storing large amounts of data. Sometimes, they even need to offset large sets of raw data to create enough space for new data. in such cases, valuable information could be lost. With Hadoop, cost-related issues do not arise, as it is a cost-effective solution for data storage Data could be stored for a long time as all the raw data generated from a company can be stored. If in the future the company decides to change their focus, necessary actions can be taken by simply referring to the old stored raw data, which ordinarily., would not be possible with the traditional approach where the data may have been deleted.

3) Speed

Every organization wants to get their tasks easily done and at a faster rate. With Hadoop, companies can easily store their data at a reasonable speed. Its file storage system involves data storage on a distributed file system. Since all the data processing tools and the data are hosted on the same server, the speed of data processing is usually faster. Therefore, huge data sizes can be processed within minutes using Hadoop.

4) Multiple copies

Data stored in Hadoop are automatically duplicated to create more copies to ensure no data loss in case of any failure. Hadoop understands the importance of stored data and ensures no data loss unless officially deleted.

Disadvantages:

1) No preventive measures

Security measures are required to be provided when handling sensitive data. The security measures in Hadoop are disabled by default. The data analyst must be aware of this and should take the necessary measures to ensure data security.

2) Small data concerns

Only a few platforms for big data functions which are not suitable for small data exists. One of such platform is Hadoop, wherein only big businesses that turn out huge data can utilize it. It is not suitable for small data analysis.

3) Risky functioning

One of the commonly used programming languages is Java. The ability of hackers to exploit frameworks based on Java has made Java the center of much criticism. Hadoop is entirely built on Java and is highly vulnerable to attacks; can also cause unforeseen damages.

Dryad

This is a large-scale framework for intensive data computing which has been developed for different applications such as batch applications and data streaming [21, 22]. While MapReduce was designed for developers to have unrestricted access (aiming for simplicity while scarifying performance), the development of the Dryad system gives the developer a complete control of the communication graph and the subroutines at its vertices [23]. The aim of developing Dryad was to migrate from the multi-core single computers to data centers with several computing units through small computer networks. All the difficult issues during the creation of a large distributed concurrent application are handled by the Dryad execution engine. Such problems are the scheduling of the computer and CPU usage, the recovering from computer or communication failures, as well as the data transportation across vertices. DryadLINQ is a collection/set of language extensions which permits large-scale programming over distributed data [24]. The data-parallel portions of a program can be automatically and transparently translated into a distributed execution plan by the DryadLINQ system. This translated version can be forwarded to the platform for Dryad execution to further ensures a reliable and efficient execution.

Kafka

The Apache Software Foundation developed Apache Kafka as an open source technology for event processing which can be used owing to its storage layer which is highly scalable and can handle a large number of real-time data feeds [25]. It can be distributed, partitioned, or replicated to improve its accuracy, and can be utilized as a system for publish-subscribe messaging. Its major features include scalability, durability, and high throughput. One of its brokers can handle several megabytes of writes and reads from several applications per second. It can be scaled up effortlessly via the addition of more nodes to a cluster. Data durability is ensured by saving data on disk and running as a collection of nodes; each node is referred to as a broker. Messages transmitted via Kafka are grouped into topics, and any application that publishes messages to a Kafka topic is referred to as a producer. Applications that subscribes to Kafka topics and processes are called consumers.

Spark

Regarding big data, Spark is the most active open source project which has attracted more attention than Hadoop. It is a frame work for in-memory cluster computing for big data analysis. It offers a simple programming interface where developers can easily use the CPU, memory, and storage resources to process large datasets across several servers. The Spark is user-friendly, multi-purpose, fast, tolerant, and scalable. The major abstraction utilized in Spark is Resilient Distributed Datasets (RDD) which can store data and tolerate faults without replication [26]. RDDs are read-only distributed shared memory [27], but during the process of machine learning[28], a number of Spark implementations is provided by Spark ships with MLlib [29] and GraphX [30] libraries, as well as the recent version of the Mahout [31]. The performance of Spark has been tested against 3 SimSQL, GraphLab, and Giraph.[31]; all were subjected to several test sets, each set having complex models of increasing size. With Spark, an application can cache data in its memory for processing, and this allows an app to reduce disk I/O. There could be a sequence of jobs in a data processing pipeline that is based on MapReduce, where each job can read data from the disk, analyze it, and write the results to the disk. Different data processing jobs can be executed parallelly on the Spark platform. It can simultaneously be used for graph computing, stream processing, interactive analysis, batch processing, and machine learning [32].

Nephele/PACT

This is a parallel system for data processing which is made up of a programming platform known as parallelization contracts, and a scalable engine for parallel execution known as Nephele [33]. It is the most important component of Stratosphere, an open-source software for parallel data processing. Nephele/PACT has the following features:

• Has a better programming model compared to MapReduce. PACT is a generalization of MapReduce but it was formulated based on output and input contracts. The input contract is a second-order function that allows a natural expression of complex data analysis operations and independently parallelizes them. The output contract is an optional component which enables certain optimizations and can annotate the properties of the first-order functions.

• Can separate the programming model from the concrete execution strategy. There are several ways to process data under PACT programs. The most efficient execution strategy can be determined by the compiler and translated into a parallel data flow program modelled as DAGs.

• Has a non-static execution engine. DAGs generated by the compiler are executed in a fault-tolerant and parallel manner. The execution process can be influenced by the PACT compiler in a fine-grained manner.

Flink

This is an open-source streaming framework which is employed for distributed, accurate, high-performing, and always-available data streaming applications. It is an open-source framework for distributed stream processing [34, 35] which provides accurate results even in the presence of out-of-order or late-arriving data. It is fault-tolerant and can recover from failures effortlessly. It can run on several nodes and still retains good throughput and latency [36]. Flink processing model applies transformation to parallel data collections [34]. Flink optimizes the user graph, taking into account the data locality. Flink uses thread-based worker model for executing the data flow graphs. It can chain consecutive tasks in the workflow in a single node to make the run more efficient by reducing data serializations and communications[37]. Flink and Spark are designed to make Hadoop scalable and fault-tolerant, and to analyze intensive data applications with distributed memory framework. They are ideal for machine learning and data mining many as the require much iterative functions, as well as their in-memory parallel execution model which saves large amount of disk I/O [38]. Such transformations reduce functions and generalize map; they also function as groups, joints or iterations. They also include a cost-based optimizer which can automatically select the best execution plan for any job. Flink can also run legacy code without modifications since it is fully compatible with MapReduce.

SAMOA (Scalable Advanced Massive Online Analysis)

The SAMOA is an open-source framework for big data mining [39, 40] which provides a collection of distributed data streaming algorithms for the commonest machine learning and data mining tasks like classification, regression, clustering, and programming abstractions during the development of new algorithms. The aim of SAMOA is to ensure the satisfaction of future big data mining demands by combining the two methods into a single open source platform [41]. It accepts input streams from S4, Samza14, and Storm. These are all Apache projects for big data stream processing. It can also build models with local testing and development files. Its algorithms are similar to those in Massive Online Analysis (MOA). ML project from the developers of Weka and the native MOA models can be used with a plugin in SAMOA. SAMOA algorithms are represented by a directed graph of operators with communicates via messages along streams that connect pairs of nodes.

Storm

This is an open source free distributed real-time computation system which simplifies the process of unbounding data stream; performing the same function for real-time processing as Hadoop did for batch processing. Storm is fun to use, simple, can be used with any programming language [8, 42]. Its development began at BackType, a social media analytics company, and was furthered at Twitter after been acquired in 2011. It was an open-sourced project which later became a top-level Apache project in 2014 [70]. They have a different architecture compared to MapReduce-based systems and employ different mechanisms for fault tolerance, message passing, and resource allocation which are specifically designed for online data streaming and processing [43]. The architecture of Storm consists of spouts and bolts; spouts are the input streams while bolts housed most of the computation logic, and responsible for data processing in the form of tuples from either the spout or other bolts. Storm was built to stand independently from Hadoop, but after Hadoop was moved to YARN, there have been efforts to combine the two frameworks. Hortonworks incorporated Storm into their Hadoop distribution starting from version 2.1, while Yahoo! is making effort to integrate them as well [44]. Any Storm processing is defined as a DAG of spouts and bolts called a topology. Storm incorporates a default scheduler which utilizes a simple scheduling solution that assigns executors to pre-configured workers in a round-robin fashion, and then, evenly assign them to available slots on the worker nodes. This scheduling strategy results in executors being evenly distributed over the available slots.

Pregel

This is a well-known graph computation model for handling big graphs of up to billion vertices and trillion edges [45]. Meanwhile, it is a low-level computation model that requires developers to write programs that need careful optimization and hard to maintain. Pregel-based structures are considered for handling such distributed graphs as they perform vertex-based computation by partitioning the processing into sub-steps and super steps [46]. It is a graph analytic platform designed for big graphs around bulk synchronous parallel (BSP) models. Other examples include graph-centric platforms such as GraphLab which rely on a graph-based “vertex programming” abstraction that opens up new opportunities for ML program partitioning, computation scheduling, and flexible consistency control; hence, they are usually correct and fast for ML [47]. A Pregel program is initiated by the loading of an HDFS-generated input graph into the machine memory [48]. A Pregel user ought to specify a user-defined function (UDF) compute (msgs) to be called by a vertex v, where msgs is the set of incoming messages received by v (sent in the previous super-steps).

MOA (Massive Online Analysis

The MOA is a software framework with advanced algorithms used for curating data streams working with different algorithms and setting to challenge practical datasets [49]. Both building and experimentation on machine learning algorithms are supported to provide quick answers to the concept of the nature of data streams [50]. Although MOA is ideal for machine learning, it is not supported on a large scale. It can only be executed on a single machine and could not be scaled to multiple machines when necessary [51]. MOA and WEKA classifier tools are different, in that, MOA classifier treats input data as data streams, though MOA is based on the WEKA classifier [52]. The processing of data streams in the MOA classifier demands fulfilling 3 main criteria: firstly, since computers have limited memory, only certain parts of the data stream can be processed within a given period; secondly, data will be assessed only once during the course of the process; finally, the process can be interrupted at any time and the classifier could still produce results up until the allocated time-based on the training results.

H2O

This is a framework for distributed machine learning and memory predictive analytics [53, 54]. It can be run either alone or with Big Data platforms such as Hadoop and Spark, where confers them with memory machine learning. At present, H2O comprises of several common machine learning algorithms, including the decision trees, gradient boosting, generalized linear models (linear regression, logistic regression, etc.), k-means, Naïve Bayes, and deep learning. In this work, energy prediction was performed using H2O deep learning [55]. It also provides a web-based user interface which makes learning tasks easier to be accessed by analysts with a weak programming background. It supports Java, R, Python, and Scala for the benefit of those that wish to tweak its implementation. It supports YARN, Spark, and some Hadoop versions. H2O can run on your desktop or scale using multiple nodes with Hadoop, an EC2 cluster, or S3 storage. H2O nodes run on Hadoop nodes as JVM invocations. To maintain the performance of the model, it is recommended that an H2O node should not be run on the same hardware as the Hadoop NameNode. Being that H2O nodes run in Hadoop as mapper tasks, they can be viewed by administrators in the normal TaskTracker and JobTracker frameworks, making the processes more visible [56]. H2O programming can be executed with Java, R, Python, and Scala. Programmers with a weak programming background can use this tool via web-based UI. Because most of the parameters in H2O comes come pre-tuned, it is easy to set up, and require less of a learning curve compared to most other free options.

Mahout

This is an open source project developed by Apache with the aim of building a standard library of data mining and machine learning algorithms [50]. Its main feature is the scalability of the algorithms in its library and its ability to perform well both in distributed and stand-alone machine environments. It can be launched on a Hadoop platform and uses other technologies such as MapReduce and HDFS. It provides significant stability and in large data management compared to other platforms that fail to do so.

Yahoo! Simple Scalable Streaming System (S4)

The Yahoo! S4 is a general-purpose, pluggable, scalable, distributed, and fault-tolerant platform that gives programmers the chance to develop programs with ease for the processing of continuous unbounded data streams [57, 58]. Although these tools focus on big data stream processing, they effectively ignore the resource scheduling strategy. Its main features are as follows:

• Decentralized: There is no centralized service as all nodes are symmetric; there is no single source of failure. This greatly reduces cluster configuration and deployment changes.
• Scalable: The addition of more nodes to the cluster linearly increases the throughput. It does not have a limit on the number to be supported.
• Extensible: Its applications can be easily written and deployed using a simple API. Its component units such as the message queues, processors, serialize, and checkpointing backend can be effectively replaced through customized implementations.
• Cluster management: All the cluster management tasks in this model are hidden using a communication layer stationed on top of the ZooKeeper. The ZooKeeper is an open-source distributed coordination service for distributed applications.
• Fault-tolerance: When there is a failure in any of its server, the standby server is activated automatically to cover for the failed server. Loss of state is minimized by checkpointing and recovery processes. More open source data platforms include R, Vowpal Wabbit, Project, PEGASUS, GraphLab, and Create™.

4.2 Vertical Scaling Platforms

In IT, vertical scaling refers to the building up of resources while horizontal scaling implies building out of resources. These are two different types of scaling that differ in their principles based on the involved software and hardware resources. A basic way of thinking about vertical scaling is the addition of extra power or capability to a single component by the administrators. A simple form of vertical scaling is the installment of more processing or memory power on a single computer. Some of the common vertical scale-up platforms are Graphics Processing Unit (GPU), High-Performance Computing Clusters (HPC), and Multicore processors, and Field Programmable Gate Arrays (FPGA). The capability of these platforms is discussed in the next sections.

High-performance computing (HPC) clusters

This is parallel computations involving several computation elements (CPU, GPU) and a fast network that connects them. Clusters are the commonest used HPC system framework. Parallel computing is more efficient when performed on several servers than on specialized systems. HPC clusters [59], also referred to as blade or supercomputers, are machines that were built with several cores. Their cache, communication mechanism, and disk organization, and are different. They use strongly built and powerful hardware that is optimized for throughput and speed. High-performance compute clusters offer the most flexible, efficient, and cost-effective HPC platform for HPC simulations.

Multicore CPU

A multi-core processor is a modern technology based on the improvements in processor and network technologies. The significant advances in CPU chips over years contributed to multi-core architectures and is a key to the processing power required for big data. It is a CPU chip architecture with dual, quad, six, or n processing cores that are all on only one CPU chip. CPU parallelism is mainly achievable through multi-threading [60] as all the cores share a common memory. The task must be broken down into threads which can be executed parallelly on different CPU cores. Most programming languages provide the libraries for the creation of threads and for a parallel use of CPU. The commonest of such languages is Java. The problem of CPU is their small number of processing cores, as well as their reliance on the memory of the system for data access. Most systems have a memory of about a few hundred gigabytes, this places a restriction on the size of data that CPUs can efficiently process. disk access is always problematic whenever the data size on the CPU has exceeded the system memory. The data may be saved in the system, but the processing speed is greatly reduced, making memory access a problem. This is avoided in the GPU using DDR5 memory instead of the slower DDR3 memory used in systems. GPU also has a high cache speed for each multi-processor, and this increases the speed of data access.

Graphics processing unit (GPU)

A GPU is a heterogeneous chip multi-processor which is highly tuned for graphics. Until recently [61], GPUs were mainly used for graphical works like image and video editing, as well as accelerating graphics-related processing. However, their parallel framework and the recent advancements in GPU hardware have resulted to the emergence of the General-Purpose Computing on Graphics Processing Units (GPGPU) [62]. Programmers can now access the GPU with ease without the need for hardware details owing to the release of the CUDA framework by Nvidia. These developments point toward an increasing popularity of GPGPU. Several fast machine learning algorithms are built on GPUs. Some libraries, like GPUMiner [63] use CUDA framework to implement few machine learning algorithms on GPU. Experimental studies have reported improvements in speed when using GPU compared to a multicore CPU. There are some problems with GPU, such as having limited memory space. Having a maximum memory space of 12 GB per GPU (as of current generation), data in the terabyte scale cannot be efficiently handled. If the data size is higher than the GPU memory, performance is significantly reduced and disk access becomes more problematic [6]. Another problem with GPU is the limited available software and algorithms. Due to the need to break down tasks in GPUs, most existing analytic algorithms are not easily compatible with GPUs.

Field programmable gate arrays (FPGA)

FPGAs were initially introduced more than two decades ago, and since then, have grown rapidly and gained more popularity in digital circuits. Technological advancements have greatly strengthened the logic capacity of FPGAs and made them a viable alternative for larger designs. Further, the programmable nature of their routing and logic resources has a significant effect on the quality of final device's area, power consumption, and speed [64]. They programing involved the use of hardware descriptive language (HDL) [65]. The cost of their development is usually higher compared to other platforms due to their customized hardware. Their software coding must be carried out in HDL with a basic knowledge of the hardware, and this increases the cost of developing the algorithm. The users must carefully investigate the suitability of any application for FPGA before implementation because they have a selective effectiveness (effective on a certain set of applications). FPGA is used as a hardware firewall and is faster when scanning large amounts of network data than software firewalls [66].

5. Summary of Process of Big Data Analytics

Several studies attempted to present an efficient or effective solution from the perspective of system (e.g., framework and platform) or algorithm level. A simple comparison of these big data analysis technologies from different perspectives is described in Table 1, to give a brief introduction to the current studies and trends of data analysis technologies for the big data. the "Name" column is an abbreviated names of the methods or platform/framework. From the analysis framework perspective, this table shows that big data framework, platform, and machine learning are the current research trends in big data analytics system. The "Description" column gives the further goal of the study. The "processing methods" column describe the way of big data analytics that will be vertical or horizontal.

Here in this paper we briefly explained the recent big data frameworks. We made comparison for the major frameworks based on the available literature. We find the bottlenecks will appear in different places of data analytics in big data, because the environments, systems and input data have changed which are different from the traditional data analytics. The data deluge of big data will fill up the "input" system and increase the computation load. One of the current solutions to the avoidance of bottlenecks is to add more computation resources while the other is to spilt the analysis works to different computation node. Also there are some problems while working with Hadoop and Spark because the complexity of them. So the solution is using Radoop framework as example. Radoop is purely visual workflow designer for for Hadoop and Spark. Radoop remove the complexity of data prep and machine learning on Hadoop and Spark. Also the combination of platforms might be more suitable for a particular algorithm and can potentially resolve the issue of making it high scalable as well as performing real-time analysis.

Last but not least, to allow the researcher in the field of big data entering this new age, we present the highest impact research trends in big data streaming field as follow:
-There is no doubt that reducing computational time using parallel computing is one of the remarkable future trends.
- How to make the machine learning algorithms operate in the parallel method because these algorithms typically do not design for parallel processing environment.
-Because of the importance of social media, so the gathering and processing the data comes from social media will be the highest impact research trend in term of big data streaming.

Table 1 - The recent big data analytics platform and methods

Name	Year	Ref	Description	Processing method
Hadoop	2011		Parallel computing platform	Horizontal
Dryad	2011		Large-scale framework for intensive data computing	Horizontal
Kafka	2011		Open source technology for event processing	Horizontal
Spark	2014		Parallel computing platform	Horizontal
SAMOA	2013		Machine learning platform for streaming data	Horizontal
Nephele/PACT	2010		Parallel system for data processing	Horizontal
Flink	2015		Open-source streaming framework	Horizontal
Storm	2014		Parallel computing platform	Horizontal
Pregel	2010		Large scale graph data analytics	Horizontal
MOA	2014		Software framework with advanced algorithms	Horizontal
Mahout	2011		Machine learning algorithms	Horizontal
YAHOO!	2010		Distributed and fault tolerant platform	Horizontal
H2O	2017		Parallel processing engine	Horizontal
HPC	1960		Machines with thousands of cores	Vertical
Multicore CPU	2001		Machine having dozens of processing cores	Vertical
FPGA	1982		Highly specialized hardware units	Vertical
GPU(CUDA)	2007		Parallel computing platform	Vertical
Radoop	2011		Data analytics and machine learning algorithms and R statistical tool	Horizontal
DBDC	2004		Parallel clustering	Horizontal

6. Conclusions and Future Work

In this paper, various platforms for data processing which are currently available were surveyed. The advantages and challenges of such platforms were discussed as well. Details of these hardware platforms, along with some popular software frameworks such as Spark and Hadoop were provided. Different platforms based on some important features such a scalability and real-time processing were compared based on ratings. This study provided a basis for the analysis of the effectiveness of the reviewed platforms, especially their strengths in handling real-world applications. Further investigations will focus on investigating the possibility of combining several platforms to solve a given application problem, such as an attempt to combine Hadoop (a horizontal scaling platform) with GPU (a vertical scaling platform). Additionally, the combination of machine learning with modern big data platforms also needs to be investigated. A combination of different platforms can be a better strategy for certain real-world problems in healthcare, IOT, smart cites, smart health applications, and big genomic datasets.

Acknowledgement

The authors would like to thank ICCI, Informatics Institute for Postgraduate Studies (IIPS_IRAQ) for their moral support. Special thanks to the anonymous reviewers for their valuable suggestions and constructive comments.

References

[1] J. Qiu, Q. Wu, G. Ding, Y. Xu, and S. Feng, "A survey of machine learning for big data processing," EURASIP Journal on Advances in Signal Processing, vol. 2016, p. 67, 2016.
[2] S. Yu, M. Liu, W. Dou, X. Liu, and S. Zhou, "Networking for big data: A survey," IEEE Communications Surveys & Tutorials, vol. 19, pp. 531-549, 2017.
[3] R. V. Zicari, "Big data: Challenges and opportunities," Big data computing, vol. 564, 2014.
[4] M. D. B. D. Explained.
[5] J. Tekli, "An overview on xml semantic disambiguation from unstructured text to semi-structured data: Background, applications, and ongoing challenges," IEEE Transactions on Knowledge and Data Engineering, vol. 28, pp. 1383-1407, 2016.
[6] D. Singh and C. K. Reddy, "A survey on platforms for big data analytics," Journal of Big Data, vol. 2, p. 8, 2015.
[7] M. Pospiech and C. Felden, "Big data–a state-of-the-art," 2012.

[8] F. Apache Storm, 2015. [Online]. http://storm.apache.org/.

[9] F. Cuda, 2015. [Online]. http://www.nvidia.com/object/cuda_home_new.html.

[10] F. Apache Hadoop, 2015. [Online]. http://hadoop.apache.org.

[11] C. J. Curtin RR, Slagle NP, March WB, Ram P, Mehta NA, Gray AG. MLPACK: a and s. C. m. l. l. J. M. L. R. 2013;14:801–5.

[12] F. Apache Mahout, 2015. [Online]. http://mahout.apache.org/.

[13] C.-W. Tsai, C.-F. Lai, H.-C. Chao, and A. V. Vasilakos, "Big data analytics: a survey," Journal of Big Data, vol. 2, p. 21, 2015.

[14] T. D. C. f. s. b. d. a. C. 2013;46(5):98–101.

[15] Z. H. Lu R, Liu X, Liu JK, Shao J. Toward effient and privacy‐ preserving computing in big data era. IEEE Netw. and 2014;28(4):46–50.

[16] S. I. Cuzzocrea A, Davis KC. Analytics over large‐ scale multidimensional data: The big data revolution!. In: and p. Proceedings of the ACM International Workshop on Data Warehousing and OLAP.

[17] H. M. W. m. f. b. d. a. a. v. I. P. o. t. I. C. o. C. S. a. E. Zhang J, 2013. pp 1021–1028.

[18] V. M. B. d. a. f. I. P. o. t. I. C. o. Chandarana P and S. Circuits, Communication and Information Technology Applications, 2014. pp 430–434.

[19] O. A. U. h. d. a. o. Apache Drill February 2.

[20] C. M. Zaharia M, Franklin MJ, Shenker S, Spark SI (2010) Cluster Computing with Working Sets. In: and p. Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing.

[21] B. S. L. a. B. H. S. Tang, "Fair Resource Allocation for Data-Intensive Computing in the Cloud," in IEEE Transactions on Services Computing, vol. 11, no. 1, pp. 20-33, Jan.-Feb. 1 2018.

[22] Y. Zhang, T. Cao, S. Li, X. Tian, L. Yuan, H. Jia, et al., "Parallel processing systems for big data: a survey," Proceedings of the IEEE, vol. 104, pp. 2114-2136, 2016.

[23] C. Dobre and F. Xhafa, "Parallel programming paradigms and frameworks in big data era," International Journal of Parallel Programming, vol. 42, pp. 710-738, 2014.

[24] Y. Yu, Isard, M., Fetterly, D., Budiu, M., Erlingsson, Ú., Gunda, P.K., Currey, J.: Dryadlinq: a system, v. for general-purpose distributed data-parallel computing using a high-level language. In: OSDI, and p. (2008).

[25] A. K. G. M. D'silva, Gaurav and S. Bari, "Real-time processing of IoT events with historic data using Apache Kafka and Apache Spark with dashing framework," 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), Bangalore, 2017, pp. 1804-1809.

[26] C. M. Zaharia M, Das T, Dave A. Fast and interactive analytics over Hadoop data with Spark. USENIX and L. 2012;37(4):45–51.

[27] K. R. I. o. T. Ni Z. Comparative Evaluation of Spark and Stratosphere. Thesis.

[28] Saba Abdul-baqi Salman , Ahmed Hussein Ali , Mohammad Khamees Khaleel, Mostafa Abdulghfoor Mohammed, "A New Model for Iris Classification Based on Naïve Bayes Grid Parameters Optimization," International Journal of Sciences: Basic and Applied Research (IJSBAR), vol. 40, pp. 150-155, 2018.

[29] M. https://spark.apache.org/mllib/.

[30] G. https://spark.apache.org/graphx/.

[31] M. http://mahout.apache.org/.

[32] M. Guller, Big data analytics with Spark: A practitioner's guide to using Spark for large scale data analysis: Springer, 2015.

[33] A. Alexandrov, M. Heimel, V. Markl, D. Battré, F. Hueske, E. Nijkamp, et al., "Massively parallel data analysis with pacts on nephele," Proceedings of the VLDB Endowment, vol. 3, pp. 1625-1628, 2010.

[34] A. J. Leich M, Schubotz M, Heise A, Rheinländer A, Markl V. Applying Stratosphere for Big Data Analy and T. a. W. B. p. 15th Conference on Database Systems for Business.

[35] A. F. https://flnk.apache.org/.

[36] T. Deshpande, Learning Apache Flink: Packt Publishing Ltd, 2017.

[37] S. Kamburugamuve, P. Wickramasinghe, S. Ekanayake, and G. C. Fox, "Anatomy of machine learning algorithm implementations in MPI, Spark, and Flink," The International Journal of High Performance Computing Applications, vol. 32, pp. 61-73, 2018.

[38] C. Chen, K. Li, A. Ouyang, Z. Tang, and K. Li, "Gpu-accelerated parallel hierarchical extreme learning machine on flink for big data," IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 47, pp. 2740-2753, 2017.

[39] A. Bifet and G. D. F. Morales, "Big data stream learning with Samoa," in Data Mining Workshop (ICDMW), 2014 IEEE International Conference on, 2014, pp. 1199-1202.

[40] L. Zhou, S. Pan, J. Wang, and A. V. Vasilakos, "Machine learning on big data: Opportunities and challenges," Neurocomputing, vol. 237, pp. 350-361, 2017.

[41] G. D. F. Morales and A. Bifet, "SAMOA: scalable advanced massive online analysis," Journal of Machine Learning Research, vol. 16, pp. 149-153, 2015.

[42] H. a. R. P. A. Mastering Apache Storm: Real-time big data streaming using Kafka, 2017.
[43] T. Li, J. Tang, and J. Xu, "Performance modeling and predictive scheduling for distributed stream data processing," IEEE Transactions on Big Data, vol. 2, pp. 353-364, 2016.
[44] E. R. Feng A, Dagit D, Roberts N. Storm-yarn. https://github.com/yahoo/storm-yarn.
[45] L.-D. Tung, "Pregel meets UnCAL: A systematic framework for transforming big graphs," in Data Engineering Workshops (ICDEW), 2015 31st IEEE International Conference on, 2015, pp. 250-254.
[46] V. Bhatia and R. Rani, "A parallel fuzzy clustering algorithm for large graphs using Pregel," Expert Systems with Applications, vol. 78, pp. 135-144, 2017.
[47] E. P. Xing, Q. Ho, P. Xie, and D. Wei, "Strategies and principles of distributed machine learning on big data," Engineering, vol. 2, pp. 179-195, 2016.
[48] D. Yan, Y. Huang, M. Liu, H. Chen, J. Cheng, H. Wu, et al., "GraphD: Distributed Vertex-Centric Graph Processing Beyond the Memory Limit," IEEE Transactions on Parallel and Distributed Systems, vol. 29, pp. 99-114, 2018.
[49] K. Poonsirivong and C. Jittawiriyanukoon, "A rapid anomaly detection technique for big data curation," in Computer Science and Software Engineering (JCSSE), 2017 14th International Joint Conference on, 2017, pp. 1-6.
[50] P. D. C. de Almeida and J. Bernardino, "Big data open source platforms," in Big Data (BigData Congress), 2015 IEEE International Congress on, 2015, pp. 268-275.
[51] W. Romsaiyud, "Automatic extraction of topics on big data streams through scalable advanced analysis," in Computer Science and Engineering Conference (ICSEC), 2014 International, 2014, pp. 255-260.
[52] D. Marrón, J. Read, A. Bifet, and N. Navarro, "Data stream classification using random feature functions and novel method combinations," Journal of Systems and Software, vol. 127, pp. 195-204, 2017.
[53] Mastering Machine Learning with R - Second Edition Paperback – April 24.
[54] L. Wilkinson, "Visualizing Big Data Outliers through Distributed Aggregation," IEEE transactions on visualization and computer graphics, vol. 24, pp. 256-266, 2018.
[55] K. Grolinger, M. A. Capretz, and L. Seewald, "Energy consumption prediction with big data: Balancing prediction accuracy and computational resources," in Big Data (BigData Congress), 2016 IEEE International Congress on, 2016, pp. 157-164.
[56] S. Landset, T. M. Khoshgoftaar, A. N. Richter, and T. Hasanin, "A survey of open source tools for machine learning with big data in the Hadoop ecosystem," Journal of Big Data, vol. 2, p. 24, 2015.
[57] F. Xhafa, V. Naranjo, and S. Caballé, "Processing and analytics of big data streams with yahoo! s4," in Advanced Information Networking and Applications (AINA), 2015 IEEE 29th International Conference on, 2015, pp. 263-270.
[58] V. Palanisamy and R. Thirunavukarasu, "Implications of Big Data Analytics in developing Healthcare Frameworks–A review," Journal of King Saud University-Computer and Information Sciences, 2017.
[59] U. Buyya R (1999) High Performance Cluster Computing: Architectures and Systems (Volume 1). Prentice Hall and N. SaddleRiver, USA.
[60] E. S. Tullsen DM, Levy HM (1995) Simultaneous Multithreading: Maximizing on-Chip Parallelism. In: ACM and p. SIGARCH Computer Architecture News.
[61] Y. Kim, J.-E. Jo, H. Jang, M. Rhu, H. Kim, and J. Kim, "GPUpd: a fast and scalable multi-GPU architecture using cooperative projection and distribution," in Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture, 2017, pp. 574-586.
[62] D. W. T. G. c. e. I. M. Nickolls J.
[63] L. K. Fang W, Lu M, Xiao X, Lam CK, Yang PY, He B, Luo Q, Sander PV, Yang K (2008) Parallel data mining on and T. R. H.-C.-. graphics processors. Hong Kong University of Science and Technology.
[64] U. Farooq, Z. Marrakchi, and H. Mehrez, "FPGA architectures: An overview," in Tree-based Heterogeneous FPGA Architectures, ed: Springer, 2012, pp. 7-48.
[65] M. P. T. V. H. D. L. Thomas DE, vol 2. Springer.
[66] R. A. Jedhe GS, Varghese K (2008) A scalable high throughput firewall in FPGA. In: Proceedings of and p. 16th International Symposium on Field-Programmable Custom Computing Machines.