Overview

Scalable machine-learning algorithms for big data analytics: a comprehensive review

Preeti Gupta,[1]* Arun Sharma[1] and Rajni Jindal[2]

Big data analytics is one of the emerging technologies as it promises to provide better insights from huge and heterogeneous data. Big data analytics involves selecting the suitable big data storage and computational framework augmented by scalable machine-learning algorithms. Despite the tremendous buzz around big data analytics and its advantages, an extensive literature survey focused on parallel data-intensive machine-learning algorithms for big data has not been conducted so far. The present paper provides a comprehensive overview of various machine-learning algorithms used in big data analytics. The present work is an attempt to identify the gaps in the work already performed by researchers, thus paving the way for further quality research in parallel scalable algorithms for big data. © 2016 John Wiley & Sons, Ltd

How to cite this article:
WIREs Data Mining Knowl Discov 2016, 6:194–214. doi: 10.1002/widm.1194

INTRODUCTION

The advent of various technologies such as Social Networks and Internet of Things has led to the generation of a global network for sharing and collaborating information. The usage and influence of digital media, especially social media, has grown in every sphere of our life. This proliferation has generated a huge amount of data, creating a data deluge. It is difficult to effectively extract useful information from all the available online information due to the volume and variety (structured/semistructured/unstructured) of data. However, the foundation of a good analytical framework relies totally on the quality of data; the richer the dataset, the better the inferences. The overwhelming amount of data necessitates mechanisms for efficient information filtering and processing. Distributed parallelism is the default choice for processing voluminous datasets as it enables faster execution of the job with optimal utilization of computational resources. It is not the case, however, that earlier parallel distributed mechanisms of processing did not exist. Mechanisms such as MPI (Message Passing Interface) were available in the past literature. However, the major advantage of big data frameworks such as MapReduce and Spark over other parallel distributed system is that it relieves the programmer from the burden of code and data distribution, replication, machine failures, and load balancing, which is taken care of by the underlying runtime system automatically. This is the main reason why big data analytics has drawn so much attention from academicians and researchers. It allows the collecting or crowding sense of data from multiple heterogeneous sources and aids in extracting useful insights more easily from the huge and heterogeneous datasets.

A large amount of research is being conducted in the evolving and multifaceted area of big data analytics. However, despite the tremendous interest in big data analytics and its advantages, an extensive literature survey focused on parallel data-intensive machine-learning algorithms for big data has not been conducted so far. This has encouraged us to provide a comprehensive survey of distributed parallel machine-learning algorithms for big data along with various optimization and performance metrics for evaluation. This is also the motivation of our paper. In the present

*Correspondence to: preeti.cool80@gmail.com

[1]Department of IT, Indira Gandhi Delhi Technical University for Women (IGDTUW), New Delhi, India

[2]Department of CSE, Delhi Technological University (DTU), Delhi, India

Conflict of interest: The authors have declared no conflicts of interest for this article.

paper, an overview of big data analytics is provided. The task of big data analytics is divided into two phases. The first phase involves setting the stage for big data processing and includes the selection of big data storage and big data computational framework. The second phase involves the designing and implementation of machine-learning tools for extracting insights from the data. The paper is divided into seven sections. The first section starts with the general introduction of the topic. The second section outlines the employed research methodology, highlighting the basis on which we selected the research papers for study and the criteria for their inclusion in our work. The third section provides an introduction to big data and big data analytics along with various challenges of big data analytics. The fourth section covers details regarding big data infrastructure. The fifth section provides details and a comparative analysis of the various distributed data-intensive parallel machine-learning algorithms for big data. The sixth section provides some limitations of our research. The seventh section presents the research implication of our research. The final section contains concluding remarks and the future scope of our work.

RESEARCH METHODOLOGY

Objective

The strong need of extracting relevant information from a data heap containing humungous and heterogeneous data demands scalable and decentralized analysis techniques. This is where big data analytics steps in. It helps researchers and data scientists in extracting useful insights from the data deluge. Traditional machine-learning algorithms need to be adapted to the big data realm for handling big data scalability. The objective of this paper is to provide a solid base for scalable machine-learning algorithms for big data analytics. The study covers an overview regarding big data storage and big data computational frameworks along with a survey of distributed parallel data-intensive algorithms for big data. Although this search may not be completely exhaustive, it provides a good comprehensive base for understanding distributed scalable machine-learning algorithms in the emergent and profound field of big data analytics. The paper also provides a comparative analysis of scalable machine-learning algorithms on various metrics.

Search Process

This paper is a qualitative review study based on the authors' perception from the study of various papers related to big data analytics. Literature selection criteria employed by authors for deciding on research paper inclusions are represented diagrammatically in Figure 1. The relevant material was scattered across various journals, conferences, and book chapters. The online databases that were referred for extracting relevant literature were IEEE, Science Direct, Hindawi ACM Digital Library, Springer, IGI global, and Taylor & Francis Online. The online databases were searched based on descriptors such as 'big data,' 'big data analytics,' 'big data challenges,' 'big data storage,' 'big data and machine learning,' 'machine learning algorithms employing MapReduce,' 'distributed parallel algorithm & MapReduce,' and 'parallel machine learning algorithms employing Spark,' which produced various journal and conference papers from the year 2008 to 2016.

The literature selection was divided into two categories. One category included the papers related to big data infrastructure, and the other included scalable machine-learning algorithms for big data. The full text of each article was reviewed to examine the applicability of the article for the review, and only the relevant articles were included. The papers that have achieved parallelism using vertical scaling were rejected. Papers that included distributed parallel programming without MapReduce or Spark were also rejected.

Key Factors for Evaluation

On the basis of the selected papers, several optimization and evaluation metrics are used to perform a comparative analysis of scalable machine-learning algorithms for big data. The optimization metrics used are sampling, indexing, intelligent partitioning, special data structure, in-memory computation, and asynchronous execution. The performance metrics used are number of MapReduce jobs, speedup, accuracy, and scalability. The performance metrics also include the data source validation check and the comparative analysis of the algorithm with the base technique, which are the effectiveness indicators of the assumption and results made in the algorithm. Scalable machine-learning algorithms were evaluated on these metrics to perform a comparative analysis.

BIG DATA ANALYTICS AND ITS CHALLENGES

Big data analytics has led to data explosion where data from sensors, devices, and networks are collected and stored without discarding anything from it. The data are flowing at very high rate, and the

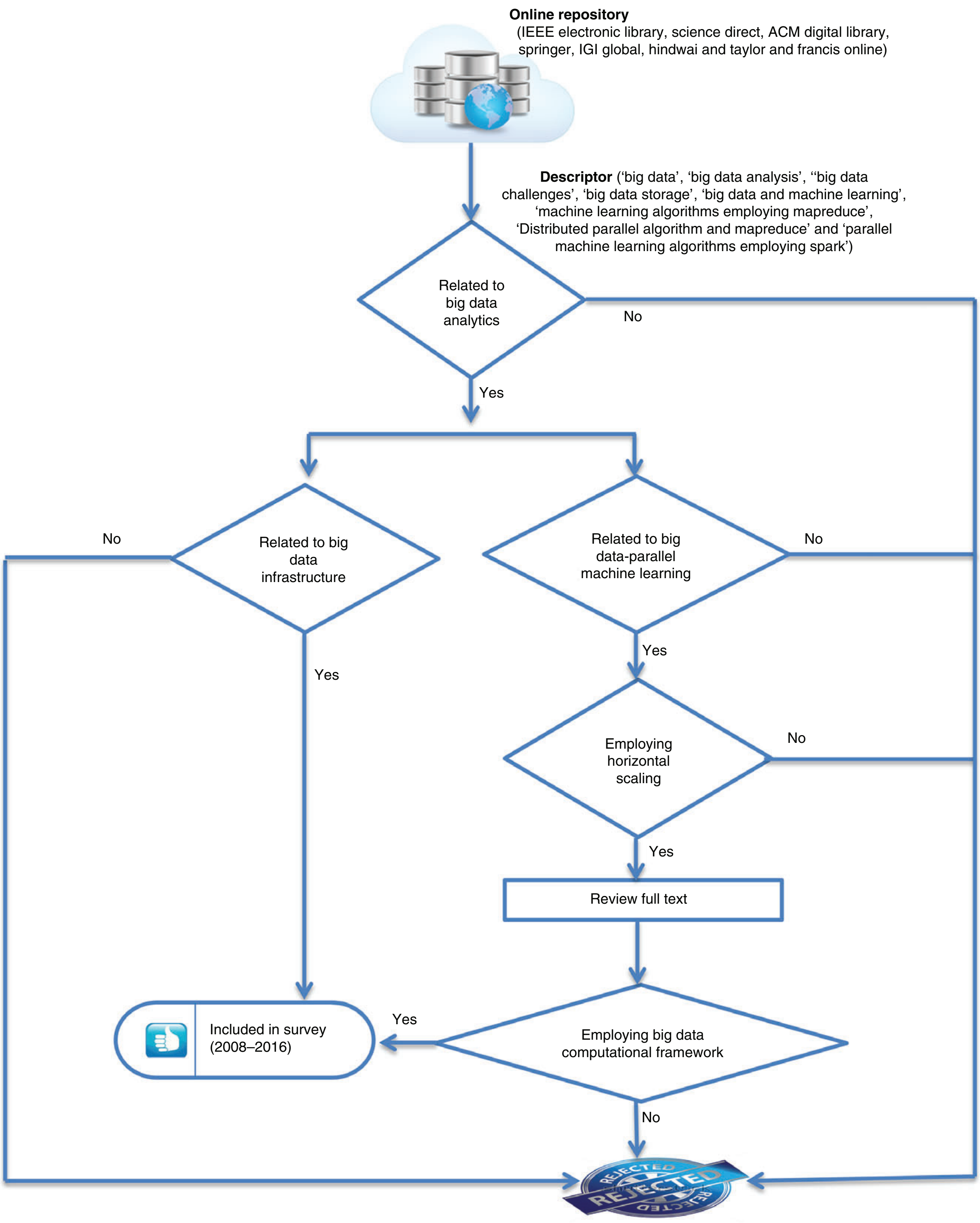
Online repository
(IEEE electronic library, science direct, ACM digital library, springer, IGI global, hindwai and taylor and francis online)
Descriptor ('big data', 'big data analysis', "big data challenges', 'big data storage', 'big data and machine learning', 'machine learning algorithms employing mapreduce', 'Distributed parallel algorithm and mapreduce' and 'parallel machine learning algorithms employing spark')
Related to big data analytics
No
Yes
Related to big data infrastructure
No
Yes
Related to big data-parallel machine learning
No
Yes
Employing horizontal scaling
No
Yes
Review full text
Employing big data computational framework
Yes
No
Included in survey (2008–2016)
REJECTED

FIGURE 1 | Literature selection process.

amount of data generated is humongous and heterogeneous in nature. These features account for the 3Vs of big data, which are Volume, Variety, and Velocity. Researchers have proposed additional features or Vs of big data. Owais and Hussein[1] have proposed the 9Vs of big data, comprising Veracity, Value, Visualization, Volatility, Variability, and Validity, in addition to the 3Vs, which refers to trustworthiness, significance, graphical representation, retention policy, consistency, and accuracy of data, respectively. These Vs must be considered while leveraging big data usage for any domain. Big data analytics of sensor log or sentiment data may help in extracting useful or significant insights from huge data, which was previously unavailable. However, sentiment data or log data may be collected from various varied sources, making it important to consider the trustworthiness, consistency, and accuracy of sources to produce correct and relevant output.

Just collecting all the data without extracting something interesting is of no use, and this is where big data analytics comes into the picture. Wu et al.[2] have defined big data analytics as the process of examining big data to uncover hidden patterns, unknown correlations, and other useful information that can be used to make better decisions. Big data analytics has been employed by many communities, such as government agencies, security, e-commerce, and health organizations.[3]

The biggest challenge in the big data era is to devise ways to analyze and visualize large datasets as just collecting big data does not make it good. Parallelism is the default choice for processing voluminous dataset as it enables faster execution of the job with optimal utilization of computational resources. Parallelism could be achieved by either vertical scaling or horizontal scaling. Vertical scaling involves increasing the processing power and storage of a single machine, whereas horizontal scaling involves the distribution of load across multiple systems. Vertical scaling could be achieved by techniques such as High-Performance Computing Clusters (HPCC), Graphics Processing Unit (GPU), and multicore processors.[4] Big data may consist of tera or peta bytes or even more data, for which vertical scaling may not be sufficient, and horizontal scaling would be the best fit. Horizontal scaling distributes the task among distributed nodes to achieve distributed processing. Traditional horizontal scaling includes a peer-to-peer network, with MPI as the communication mechanism.[4] However, due to self-managing and a fault-tolerant file system, the default data store for big data is the Hadoop Distributed File System (HDFS).

Big data scalable infrastructure needs to be augmented with machine-learning tools in order to extract insights from the data sea. Big data is thus broad spectrum comprising evolving techniques and tools in order to handle data-intensive jobs. Traditional algorithms need to exploit parallelism in order to efficiently handle big data problems. Traditional data analysis algorithms need to be adapted to the big data environment to make them efficient for handling high-volume, heterogeneous, and dynamic data. The major challenges imposed by big data on data analytics are[5,6]:

(**a**) Scalability: Big data collected from various sensors would be voluminous and evolving in nature. The data analysis algorithm for big data thus needs to take into account large or complex datasets. They should be scalable enough to handle a growing amount or velocity of data gracefully.

(**b**) Decentralization: The traditional data analysis was designed with a view that they will be run on a single machine on a limited dataset. However, sensor data may be spread across multiple machines or clusters, for which centralized solutions will not work. Thus, big data algorithms need to be distributed in order to handle the voluminous nature of big data.

(**c**) Dynamic datasets: The traditional algorithms were designed to operate on static data and were not able to handle data on the fly or streaming data. However, sensor data are evolving in nature and are generated at a very fast rate, for which real-time analysis may be required. Big data algorithms need to operate on a dynamic dataset and may also require the learning process to be fast in order to carry out real-time analysis. A dynamic dataset may also need to exploit a dynamic graph for visualization, projecting real-time scenarios.

(**d**) Nonuniform Dataset: The dataset for big data may come from multiple disparate sources comprising unstructured and heterogeneous data. It may involve a collection of data from various sources, such as web logs, social media, or sensor data, which will have different data formats. Thus, big data algorithms need to handle nonuniform datasets with a lot of noise also.

(**e**) Privacy: Big data analysis has been widely adopted in various industries such as healthcare due to its potential in harnessing relevant insights. However, big data has a high volume of data collected from various sources that raises more privacy concerns for the people involved in the data. The traditional methods for protecting data privacy will not work on dynamic and huge big data.

In order to handle these challenges, the realm of big data analytics in this paper has been divided into two phases, big data infrastructure and big data inference engine, as shown in Figure 2. The first phase involves setting the stage for big data processing and involves the selection of big data storage and big data computational frameworks, which have been discussed in detail in the fourth section. The second phase involves the designing and implementation of machine-learning tools in order to extract insights from the data sea, which have been discussed in detail in the fifth section.

BIG DATA INFRASTRUCTURE

Big data collection, storage, and analysis demand data decentralization and provide a new strategy for the traditional data warehousing approach. It fits into the MAD model.[7] MAD refers to the Magnetic, Agile, and Deep analysis of data. Big data allows the capture and storage of anything as is, without any transformation, acting as a magnet attracting any kind of data. It needs agile infrastructure that is fully scalable to handle data evolution of a varied nature. It needs to provide a deep data repository with deep analytical capabilities. It has not replaced Data Warehouse; instead, it has provided an edge to the overall strategy by offering schema on query rather than schema on design.[8] The Extract, Transform, and Load (ETL) approach has been transformed into ELT, i.e., Extract, Load, and Transform.[7]

Consequently, a big data infrastructure needs distributed file system (DFS), Not Only SQL (NoSQL) databases, computational frameworks providing storage, and analytical scalability.

Distributed File System

DFS provides scalable data storage and access for handling big data. DFS enables parallel processing on a large amount of data. HDFS has become the default choice for big data tasks due to its fault tolerance and its ability to be deployed on low-commodity hardware.[9] HDFS allows users to store data in files supporting a wide variety of data. The files are further split into blocks. The data blocks are distributed and replicated across nodes, providing fault tolerance.[10] HDFS supports write-once-read-many semantics on files.[9] The data block size is usually 64 MB and is replicated into three copies by default.[11] A HDFS cluster operates in a master–slave architecture.[9] It consists of one master NameNode and a number of DataNodes. NameNode manages the filesystem namespace and controls access to the files. NameNode also maps the data blocks to DataNodes. DataNode creates stores and retrieves data blocks as per the NameNode's direction.[11]

HDFS is not a database; it is just a filesystem. HDFS is provided as a part of the Hadoop framework. Hadoop is a library of tools for big data. It consists of Hadoop Common Utilities, HDFS, and MapReduce. After the introduction of Hadoop 2.0, YARN (Yet Another Resource Negotiator) has been added to the Hadoop framework.[11] YARN has separated the resource management functions from the programming mode, which has brought significant

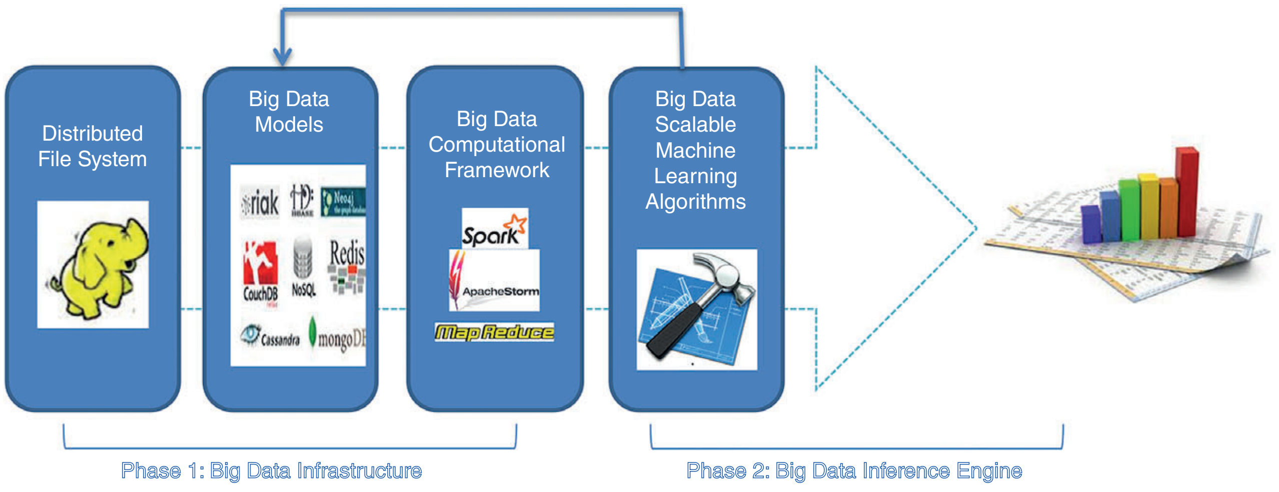
Distributed File System
Big Data Models
riak
HBASE
Neo4j
Redis
CouchDB
NoSQL
Cassandra
mongoDB
Big Data Computational Framework
Spark
ApacheStorm
MapReduce
Big Data Scalable Machine Learning Algorithms
Phase 1: Big Data Infrastructure
Phase 2: Big Data Inference Engine

FIGURE 2 | Big data analytics.

performance improvements such as support of other processing models besides MapReduce.[11]

A number of other alternatives of HDFS are also available, such as Ceph, Tachyon, GlusterFS, QFS, GridGain, and XtreemFS.[12]

Data Models for Big Data

Big data has led to an explosion of databases due to the inability of traditional databases to scale-up with big data. The big data models could be categorized into NoSQL and NewSQL data models, which could be used in place of, or along with, traditional RDBMS. CAP theorem states that any networked shared-data system can have, at most, two of the three desirable properties of consistency, high availability of data, and tolerance to network partitions.[13] A distributed system cannot lose on tolerance to network partitions, leaving a choice between consistency and availability.[13] RDBMS systems are more focused on consistency rather than availability and strictly comply with ACID (Atomicity, Consistence, Isolation, Durability) properties. NoSQL, on the other hand, are focused on availability rather than consistency and comply with BASE (Basically Available, Soft-state, Eventually consistent) properties. RDBMS handles both data storage and data management, whereas NoSQL databases separate both. NoSQL databases focus only on high-performance scalable data storage, and data management tasks are handled by the application layer.[10] NoSQL supports schema-free storage or dynamic changing schema, enabling applications to upgrade table structure without table rewriting.[10]

NoSQL are horizontal scalable databases that can handle a variety of big data and are categorized into four models. They are key-value store, wide-column store, document-oriented store, and graph database.[14,15]

(**a**) Key-value store: These systems store data in key-value pairs where value contains data that could be retrieved with the help of key. The value may be a string or complex lists or sets. The search can only be performed against keys and is limited to exact matches. The key-value store is suitable for the fast retrieval of values, such as retrieving product names or user profiles. Some of the well-known implementations of a key-value store are Dynamo, Voldemort, Redis, and Riak.

(**b**) Wide-column store: These systems employ a distributed, column-oriented data structure that accommodates multiple attributes per key. The wide-column store is suitable for distributed data storage, large-scale batch-oriented data processing, and exploratory and predictive analytics. Some of the well-known implementations of wide-column store are Bigtable, Cassandra SimpleDB, and DynamoDB

(**c**) Document-oriented store: It stores the data as the collections of the documents. These documents are encoded in a standard data exchange format such as XML, JSON, or BSON. The value column in a document store contains semistructured data and may contain a large number of attributes. The number and type of these attributes may vary among rows. Document store allows searching through both keys and values, unlike a key-value store, which allows searching only by key. Document stores are good for the storage of literal documents such as text documents, email messages, and XML documents. They are also good for storing sparse data. Some of the well-known implementations of document-oriented store are CouchDB(JSON) and MongoDB (BSON).

(**d**) Graph data model. These models use structured relational graphs of interconnected key-value pairs. Data is stored in the form of a graph having nodes and links, with attributes associated with them. Graph data models are useful when relationships between data are more important than the data itself, such as social networks and recommendations. Some of the well-known implementations of graph data store are Neo4j, InfoGrid, and GraphDB.

NewSQL is another type of big data model and is known as the next generation scalable relational database management systems.[16] NewSQL implements the best of both traditional databases and NoSQL databases. NewSQL is based on relational model and provides scalability of NoSQL while still maintaining ACID properties.[16] NewSQL supports SQL query. However, the underlying architecture of NewSQL is different from RDBMS and supports scaling techniques, such as vertical partitioning, horizontal partitioning, or sharding and clustering.[16] Some well-known implementations of NewSQL are NuoDB and VoltDB.

These data models differ significantly from each other and may be selected based on the underlying task.

Big Data Computational Frameworks

There are a number of parallel distributed computation models for big data. The choice of these big data models depends mainly on the underlying domain or application. These parallel processing models could be further categorized into batch mode, real time, and hybrid mode.[17]

Batch Mode

In this mode, the data collected are divided into batches and processed in parallel by multiple nodes. MapReduce is a distributed framework which provides data parallelism for processing large datasets and is the best fit in this category. MapReduce is based on shared nothing architecture where each processing node is self sufficient to carry out its task. Map and reduce are its two main operators. The input dataset is divided into data chunks that are operated by the mapper to compute intermediate values. Similar intermediate values are grouped based on the common key. The reduce function then takes these grouped values to produce an aggregated result. MapReduce fits into SIMD architecture suitable for embarrassingly parallel problems. MapReduce jobs take the input from a disk, and all the results, including intermediate results, are written to disk every time, incurring a significant overhead for iterative jobs in terms of I/O cost and network bandwidth.[18] MapReduce's major advantage over other parallel distributed systems is that it relieves the programmer from the burden of code and data distribution, replication, machine failures, and load balancing, which is taken care by the underlying runtime system automatically. Hadoop is the open-source implementation of MapReduce, which also provides its own DFS, HDFS. Doulkeridis and Nørvåg[19] have pointed out various weaknesses and limitations of MapReduce, such as high communication cost, lack of iteration, and lack of real-time processing. The popularity of MapReduce has now faded mainly due to a high overhead for iterative tasks in machine learning.[20]

Several big data projects have been proposed in literature that addresses the shortcomings of MapReduce while still maintaining scalability and fault tolerance capability, such as Incoop, IncMR, Haloop, Twister, iMapReduce, Continous MapReduce, Shark, Dremel, and Scalding.[17,19]

Real-Time Processing

Real-time processing involves processing data or stream as it arrives and produces output almost in real time. The programming model thus employed for batch processing is not suitable for real-time processing. Storm is the most deployed computation system for the real-time processing of unbounded streams with a very low processing latency.[17] Storm architecture consists of topologies that are the network of spouts and bolts. A spout constitutes a source stream, such as input from a Twitter stream, whereas a bolt consists of processing and computational logic, such as joins and aggregation.[20] The major difference between storm topologies and MapReduce jobs as identified by Sundaresan and Kandavel[17] is that storm topologies keep on executing due to real-time date ingestion of stream data, whereas MapReduce jobs eventually finish.

Various other storm alternatives, such as Trident and S4, have also been proposed in literature.[17]

Hybrid Model

A hybrid model can handle both batch data and real-time data. Spark is the most deployed big data hybrid processing model. It does not have its own file system and usually runs on top of HDFS. It employs in-memory computation, reducing read and write disk operations. The main building block in Spark is the Resilient Distributed Dataset (RDD), which provides in-memory data caching of intermediate results and fault tolerance without replication.[21] Spark also offers microbatch streaming in which an incoming stream is divided into different chunks to be processed by the batch system.[22]

A comparison of these three big data computation models is presented below in Table 1.

All the computation models mentioned above are parallel and have their own pros and cons. The decision to select the best model depends on the domain/application and processing time, i.e., latency. At one end is MapReduce, which is a batch-processing framework, and on the other end is Storm, which is a real-time processing framework. Marz and Waren[23] have proposed a lambda architecture that combines different technologies. It combines the process of batch analytics and real time stream analytics. Lambda architecture decomposes the problem into three layers, as shown in Figure 3. The three layers are batch layer, speed layer, and serving layer. The data are sent both to the batch layer and the speed layer as soon as it arrives. The batch layer appends the new data to the already existing dataset and performs the computation on it, which is then passed to the serving layer for indexing. The speed layer compensates the latency in the output of batch layer by providing real-time updates of the new data. The service layer merges the output of both layers to provide the federated view. Hybrid models such as Spark fit well into lambda architecture, seamlessly integrating both the modes. Stream processing of click streams followed by batch processing for the correlation of clicks is one such example. Vizard[24]

TABLE 1 | Comparison of Big Data Computation Models

	Batch Processing	Real Time	Hybrid (Batch + Real time)
Popular big data parallel model/framework	MapReduce	Storm	Spark
Data processing model	Data parallel batch processing model	Real-time task parallel computation model	Data parallel generalized batch processing model with microbatch streaming
Big data data structure	Key-value pairs having functions	Topology consisting of spouts and bolts	RDD, data frames
Big data dimensions	Volume	Velocity	Volume, velocity
Characteristics	Fault tolerant, scalable, latency of few minutes	Fault tolerant, scalable, latency for few subseconds, suitable for real-time processing	Fault tolerant, scalable, in-memory computation, latency of seconds, suitable for near real-time task, RDD makes them suitable for iterative tasks
Limitations	Selective access to data, high communication cost, lack of iteration, lack of interactive, or real-time processing	Lack of iteration and graph processing support	Selective access to data, Lack of interactive or real-time processing, Insufficient main memory may result in reduced performance
Applications	Forensics, log analysis, indexing	Online machine learning, e-commerce, Security event monitoring	Recommendation (Ad Placement), Fraud Detection

RDD, resilient distributed dataset.

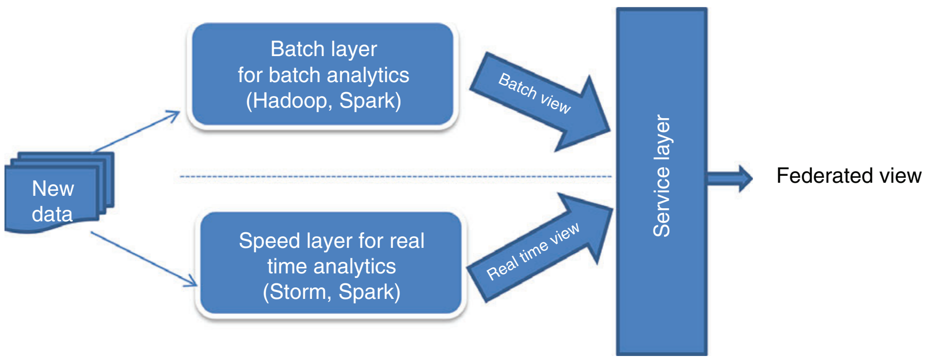

FIGURE 3 | Lambda architecture.

has pointed out that Spark has almost replaced MapReduce and has become the default standard.

BIG DATA INFERENCE ENGINE

Big data scalable infrastructure needs to be augmented with machine-learning tools in order to extract inference and insights from the data sea. The machine-learning algorithms for big data are required to fit into the distributed parallel programming paradigms for handling the inherent scalability of big data. Parallel and distributed techniques have been used for a long time for scalability and speedup. However, they all suffer from memory and processor distribution challenges. MapReduce provides the automatic handling of data distribution, load balancing, and fault tolerance, allowing easier and faster scalability of parallel systems.

Scalable Machine-learning Algorithms for Big Data

Several parallel machine-learning algorithms are proposed for big data, which employs MapReduce or other variants of MapReduce. This section presents a literature review of data-intensive parallel machine-learning algorithms to see how they are using MapReduce or Spark to achieve the desired objective. Several articles have also achieved data parallelism with the help of MapReduce and GPUs. However, the use of GPUs may not be fully scalable to big data, due to which they are not considered in the literature review.

The algorithms are categorized into various machine-learning classes, which include unsupervised learning, supervised learning, semi-supervised learning, and reinforcement learning, including deep learning.

Unsupervised Learning

Unsupervised learning finds hidden structure in unlabeled data. Clustering is one of the important forms of unsupervised learning. Clustering partitions the datasets into clusters or groups such that intracluster similarity between the data points is maximum, and intercluster similarity is minimum. Clustering has a number of applications, such as customer segmentation, document retrieval, image segmentation, and pattern classification. Several clustering algorithms have been proposed for the MapReduce framework. A brief description of some of them is presented below.

K-means clustering has been one of the most popular clustering algorithms. Zhao et al.[25] proposed a parallel version of the k-means (PK-means) algorithm based on MapReduce. Distance calculation is the most expensive task in the k-means algorithm and could be executed independently for various data points in parallel. Authors have achieved speedup by employing MapReduce for parallelism. PK-means uses three functions, map, combine, and reduce. The map function assigns each data point to its closest center. The combine function then aggregates the intermediate results of the map function and passes it to the reducer. The reduce function updates the centroids. This MR cycle is performed iteratively until termination condition has been met. A number of MapReduce (MR) jobs would involve immense I/O activity, which could be an issue for large datasets. Thomas and Annappa[26] have also employed parallel k-means MapReduce clustering algorithms for the prediction of optimal paths in self-aware Mobile Ad-Hoc Networks.

The biggest drawback in k-means is the selection of initial centroids. K-means++[27] is a modification of k-means that selects subsequent centroids (after the first centroid, which is selected at random) based on probability proportional to overall error. Bahmani et al.[28] have proposed a scalable k-means++ (K-means||) on MapReduce. K-means|| samples k points per iteration with the help of mappers in parallel, which are sent to reducers. The process is repeated for O(log n) iterations, resulting in O(klogn) points as candidates. These candidates' points are used to form k clusters using k-means++. Authors have showed that k-means|| leads to faster convergence time for iteration.

Density-based clustering algorithms find the clusters based on the region of density. It identifies dense clusters of points, allowing it to learn clusters of arbitrary shapes, and helps in identifying the outliers. Unlike k-means, they do not need to know the number of clusters in advance. DBSCAN is a well-known density-based clustering algorithm and has been employed in a number of applications, such as image processing pattern recognition and location-based service. He et al.[29] have proposed MR-DBSCAN, a MapReduce version of DBSCAN. MR-DBSCAN consists of three stages. In the first stage, the dataset is partitioned based on computational cost estimation to achieve load balancing for heavily skewed data. In the second stage, local clustering, i.e., sequential DBSCAN is performed in parallel by mappers on all partitions. In the third stage, the partial results generated are aggregated by reducers to generate global clusters. Cludoop is also a density-based clustering algorithm proposed by Yu et al.[30] that incorporates CluC as the serial clustering algorithm utilized by parallel mappers. CluC utilizes the relationships of connected cells around points, instead of an expensive completed neighbor query, significantly reducing the number of distance calculations.

Hierarchical clustering aims to make a hierarchy of clusters either using divisive clustering, which is a top–down approach, or agglomerative clustering, which is a bottom–up approach. Single-linkage hierarchical clustering (SHC) is an example of agglomerative clustering. Achieving parallelism for such an algorithm is tricky due to inherent data dependency. Jin et al.[31] have proposed a Spark-based parallel SHC algorithm, SHAS, which obtains a Minimum Spanning Tree (MST) for achieving clustering. The Divide and Conquer strategy is adopted by SHAS, which divides the original dataset into subsets. MSTs are calculated locally for each of these subsets. These intermediate MSTs are combined by employing the K-way merge until a single MST is left. SHAS uses a special data structure(union-find), which records the information about each vertex and is used to efficiently combine these partial MSTs. Authors have compared the performance of the SHAS with its MapReduce version. They have figured out that Spark performs better than MapReduce for all kinds of datasets due to in-memory computation and no overhead in setting and tearing of jobs for each iteration.

Spectral clustering relies on Eigen decompositions of affinity, dissimilarity, or kernel matrices to partition data into clusters.[32] It has been employed in various applications, such as speech recognition and image processing. Power iteration clustering (PIC) reduces the computational complexity of spectral clustering.[32] PIC replaces the Eigen decomposition of the similarity matrix required by spectral clustering into a small number of iterative matrix–vector multiplications. Yan et al.[32] have proposed a parallel power iteration clustering (p-PIC) using MPI for handling large data, which was further implemented by Jayalatchumy and Thambidurai[33] on MapReduce due to its better fault tolerance ability. MapReduce is used for affinity matrix calculations in parallel with the help of mappers, the results of which are globally aggregated by the reducer. Authors have showed that the accuracy in producing the clusters is almost the same as using a MapReduce framework.

SUB-CLU is a subspace clustering algorithm that aims to finds clusters hidden in subsets of the original feature space by using the DBSCAN algorithm and is employed in high-dimensional datasets. Subspace clustering has been employed in a number of applications, such as sensor network, bioinformatics, network traffic, image processing, and compression. Zhu et al.[34] have proposed CLUS, which has implemented a subspace clustering algorithm on top of Spark to achieve scalability and parallelism. CLUS speeds up the SUB-CLU algorithm by executing multiple density-based clustering (DBSCAN) tasks in parallel. DBSCAN is performed initially on one-dimensional subspace followed by higher-dimensional subspaces, excluding noise from lower subspaces. The process ends when no higher-dimensional subspace could be generated. It also employs a data partitioning to induce data locality. Cordeiro et al.[35] have proposed another subclustering algorithm on MapReduce, BoW (Best of both Worlds), that automatically spots bottlenecks and picks a good strategy to balance the I/O and network cost. It can employ most of the serial clustering method as a plug-in. Authors have proposed two methods, ParC (Parallel Clustering) and SnI (Sample and Ignore). ParC performs data partitioning and then finds clusters using MapReduce by employing any serial clustering algorithm. SnI performs sampling first to get rough clusters and then clusters the remaining points using ParC. ParC is executed on the whole dataset only once, minimizing disk access. SnI performs two MR jobs to reduce communication cost at the expense of higher I/O cost. ParC and SnI have their own merits, the selection of which is determined by a cost-based optimization technique. BoW calculated the cost of both algorithms on the basis of parameters such as file size, network speed, disk speed, start-up cost, and plug-in cost. The calculated cost is then used to select the best algorithm among the two. However, BoW is a hard clustering method, which may not work well for overlapping clusters.

Coclustering, or bi-clustering, or simultaneous clustering is an unsupervised learning technique that simultaneously clusters objects and features, or in other words, it allows parallel clustering of the rows and columns of a matrix. In this case, clustering is performed row-wise, keeping column assignments fixed to find the best group. The same process is executed, in parallel, column-wise, keeping row assignments fixed. It has been employed in a number of applications, such as text mining, collaborative filtering, bioinformatics, and graph mining. Papadimitriou and Sun[36] have proposed a distributed coclustering model, DisCo, with MapReduce. In MapReduce distributed processing, initially, global parameters consisting of adjacency matrix, row vector, and column vector are formed and broadcasted. Then, mappers perform the local clustering by reading rows, which are fed to the reducer to perform global clustering. The process is repeated till the cost function stops decreasing (the same process is also followed for column-wise).

Supervised Learning

Supervised learning infers a function from labeled trained data, which are used further for verification and classification. The two broad subcategories in supervised learning are classification and regression.

(**a**) Classification: Classification is a supervised learning technique that is used to determine the class of variables or to predict future trends. The classification algorithm usually consists of a training phase to construct a training model and a testing phase for the verification. Classification has a number of applications, such as spam filtering, image recognition, speech recognition, text categorization, and fraud detection.

(**b**) Regression: Regression is also a supervised learning technique used for prediction. Regression predicts a value from a continuous dataset, whereas classification is used with a categorical dataset. The variable to be predicted is known as the dependent variable. There is only one dependent variable in regression. The variables used for modeling or training are known as independent variables. If the number of independent variables is one, then, it is known as linear regression, and if it involves multiple independent variables, it is known as multiple regression.

A brief description of some the classification and regression algorithms adapted to the MapReduce framework are presented below.

A decision tree generates the classification or regression rules by recursively partitioning the dataset. Decision trees with categorical target values are known as classification trees, whereas where target values are continuous, they are known as regression trees. A decision tree consists of a root, internal node or decision node, and leaf nodes representing class labels. The decision node represents a test on the attribute, with edge representing the test condition, creating partitions of the dataset. The decision rule constitutes the path from root to leaf node. Decision tree classification consists of a training phase that is used to build the tree in a recursive manner, with an objective to optimize the classification rule. In the test phase, a decision tree is used to determine the class to which a new instance belongs. A decision tree has been employed in a number of applications, such as spam filtering, text categorization, and fraud detection. C4.5, an extension of ID3 algorithm, is a well-known decision tree that uses information gain ratio for splitting attributes. Dai and Ji[37] have proposed C4.5 using a MapReduce programming model, MRC4.5, which transforms the traditional algorithm into a series of Map and Reduce procedures. The data is vertically partitioned to maximally preserve localization characteristics. Before starting the algorithm, data preparation is performed to design some data structures, such as attribute table, count table, and hash table, to minimize the communication cost. After data preparation, the best splitting attribute is decided using a single MR job. After selecting an attribute, the training dataset is partitioned into several parts such that similar value records values are in the same partition preserving benefits of vertical partitioning. MR jobs are executed on each partition recursively to determine the best splitting attribute at the current node until the complete partition is assigned to the same class. Linkages between nodes are created to build a decision tree, which is used for the classification of new instances. Only map function is needed for the testing phase. Authors have indicated that the MapReduce implementation of the C4.5 algorithm exhibits both time efficiency and scalability. Similarly, Purdila and Pentiuc[38] have proposed MR-tree, a MapReduce version of ID3. Salperwyck et al.[39] have proposed CurboSpark, a Spark version of their time-series decision tree, i.e., CurboTree. Authors have used hierarchical clustering for the generation of the best-split attribute followed by a decision tree for classification. Authors have also tested the framework to understand and draw analytical insight about the electrical consumption of smart meters. Authors have indicated that the use of Spark library has resulted in significant improvement due to in-memory computation.

Random forest is based on ensemble learning and is used for classification or regression. It builds a number of decision tree on subsets of training data, which together creates a classification model. The classification model so formed is used for predicting the class of samples. Han et al.[40] have proposed SMRF, a scalable random forest algorithm based on MapReduce. It has two phases. The first phase is used for building the classification model, and the second phase is used for verification or prediction. In the training MR phase, data is partitioned among nodes that are operated by the mappers to create decision trees. The outputs of mappers are simply combined by the reducers or combiner. All decision trees from mappers create a forest or classification model, which is used in next phase for prediction. Similarly, PLANET[41] and MReC4.5[42] are also based on ensemble-based approaches for scalable tree learning on large data using MapReduce. Singh et al.[43] have employed random forest for peer-to-peer botnet detection on MapReduce.

Fuzzy Rule-Based Classification Systems (FRBCSs) are effective classification systems and can handle uncertainty, vagueness, or ambiguity. They are used in applications such as pattern recognition and classification. Rio et al.[44] have extended the Chi-FRBCS, a linguistic fuzzy rule-based classification system, to big data, Chi-FRBCS-BigData, which uses the MapReduce framework to learn and fuse rule bases. It uses two MR jobs for the algorithm. One MR job is used to build the model from a big training set. The other MR job is used to estimate the class of the big data sample set using an earlier model. The Chi-FRBCS-BigData algorithm has been developed in two different versions, Chi-FRBCS-BigData-Max and Chi-FRBCS-BigData-Ave, which may be selected on the basis of speed and accuracy trade-off. The Ave version provides better accuracy, whereas the Max version provides faster results. Authors observed a reduction in classification accuracy when using a larger number of maps as the rule weights are calculated from smaller data partitions.

Naive Bayes is a supervised classification technique based on statistical Bayes' theorem with independence assumptions among variables. The Naive

Bayes approach uses joint probability to estimate the probabilities over a set of classes. It has been employed in applications such as text classification, spam filtering, and sentiment analysis. Dai and Sun[45] have implemented the Naive Bayes text classification algorithm based on a rough set using MapReduce. The authors have combined a rough set theory in soft computing with Naive Bayes to improve its limit of independence, reducing bias. It selects a closet approximate independent attributes subset. In the training phase, an MR job is used to normalize the weighted frequency of each key. The testing phase is used to predict the output of test records. Liu et al.[46] have also implemented a MapReduce version of the Naive Bayes classifier for sentiment analysis.

Support vector machines (SVM) are a classification and regression model that attempts to find the best possible surface to separate classes in training samples. The learning ability of SVM is independent of the dimensionality of feature space, which makes them a suitable candidate for achieving parallelism with MapReduce. LibSVM is one of the most widely adopted SVMs. Sun and Fox[47] have proposed a Parallel LibSVM on MapReduce to improve the training time. In this model, training samples are distributed into subsections, which are trained in parallel by mappers using LibSVM. Support vectors from the map jobs are collected by the reducer and fed back as input until all sub-SVMs are combined into one SVM. Based on the parallel SVM model on MapReduce (PSMR), Xu et al.[48] have proposed an e-mail classification model using the Enron dataset. Similarly, Khairnar and Kinikar[49] have proposed sentiment analysis-based mining and summarizing using parallel SVM using MapReduce (MRSVM).

Artifical neural networks (ANNs) are widely adopted for classification and regression tasks. The back-propagation neural network (BPNN) is the most commonly encountered ANN. BPNN is a multilayer feed-forward network. It adjusts the classification parameters by employing an error back-propagation (BP) technique until its parameters are attuned to all input instances.[50] The PBPNN proposed by Liu et al.[50] presents a parallel version of BPNN using MapReduce and Spark, which distributed the iterations and computation of BPNN in parallel to achieve speedup. PBPNN distributes the data to mappers for parallel processing. Each mapper locally tunes subsets of data using BPNN iteratively. The outputs of all mappers are then ensemble by the reducer or joiner to generate classification rules. Classification accuracy is maintained by merging weak classifiers into a strong classifier with the help of ensemble techniques such as bootstrapping and majority voting. Bootstrapping helps in maintaining original data information in data subsets, and major voting helps in generating a strong classifier. The authors have performed the experiment on MapReduce, Haloop, and Spark computing frameworks. It was found that among all, MapReduce performs the worst, with a slight improvement in the case of Haloop, and Spark performs the best due to in-memory computation. Similarly, Zhang and Xiao[51] have implemented a parallel multilayered neural network employing a BP training algorithm on MapReduce (MRBP) on cloud computing clusters for speedup.

Multiple linear regression is a widely adopted regression technique that has a very high training time and may fail in case of a large dataset. Rehab and Boufarès[52] have proposed multiple linear regression on MapReduce (MLR-MR) for speedup and scalability over a large dataset. MLR-MR distributes the QR decomposition and ordinary least squares computation in parallel using MapReduce to estimate the relationship between the predictor and the target variables. MLR-MR has three steps. In the first step, the matrix is divided into small blocks. In the second step, mappers compute a local QR factorization for each such block, and in the third step, the results of all mappers (Qi results) are aggregated to have final coefficient results. Authors have showed that MLR-MR is able to train large data and effectively solves the out-of-memory problem.

Semisupervised Learning

Semisupervised learning is used where the data contains small labeled data and large amount of unlabeled data. It uses labeled data and unlabeled data for training. Semisupervised learning is well suited to big data in which majority of data may be unlabeled, and the cost of labeling all data may be too high. Cotraining and active learning are the two important techniques employed for semisupervised learning.

Cotraining is a semisupervised learning technique in which each example is partitioned into two distinct views and where two views are independently sufficient. It learns to separate classifiers for each view, and then, the most confident prediction of each classifier on unlabeled data is used for labeled trained data. Hariharan and Shivshankar[53] have presented a cotraining-based multiview learning algorithm under semisupervised conditions on MapReduce. The authors have also proposed the use of data structures such as mapping table, label file, and bidirectional reducers in order to avoid broadcasting of new labels to all nodes in the MapReduce computation. A mapping table contains partition details, and a label file contains the labels of data points in each partition. Bidirectional reducers update the labels in the label file directly. Classifiers on multiple views are learnt locally using the mappers on each node individually, and the results are combined to generate a global model using a bidirectional reducer.

Active learning is another semisupervised learning technique in which learning algorithms interactively query the source for a certain type of instance to obtain output, reducing learning effort. Zhao et al.[54] have proposed a Punishing-Characterized Active Learning Back-Propagation (PCAL-BP) neural network algorithm for big data to improve the learning efficiency of the BP model.

Reinforcement Learning

Reinforcement learning continuously learns in an environment by trial and error. It discovers the actions that yield the greatest rewards to make better decisions. The Markov Decision Process and Q-Learning are two well-known examples of reinforcement learning. Reinforcement learning has been employed in applications such as gaming and robotics.

Li and Schuurmans[55] applied the MapReduce framework for methods such as policy evaluation, policy iteration, and value iteration by distributing the computation involved in large matrix multiplications. Parallelism was used to speedup large matrix operations but not to parallelize the learning. To the best of our knowledge, the work of Li and Schuurmans[55] is the only paper that has used MapReduce for reinforcement learning of linear representations. Parallel reinforcement learning in which agents learn in parallel to achieve speedup convergence such as Bayesian Reinforcement Learning[56] are being proposed in literature.

Deep Learning

Deep learning refers to machine-learning techniques that use supervised and/or unsupervised strategies to automatically learn hierarchical representations in deep architectures for classification.[57] Deep learning has been employed in a number of applications such as image search, threat detection, fraud detection, sentiment analysis, and log analysis. A Deep Belief Network (DBN) with Restricted Boltzman Machine (RBM) is one of the most widely used frameworks for deep learning. RBMs are used to generate a training model in an unsupervised manner, which is used by DBN subsequently for supervised classification or fine tuning. Zhang and Chen[58] have proposed a distributed learning paradigm using MapReduce. Pretraining is achieved by distributed learning, which involves stacking a series of distributions of RBMs followed by distributed BP for fine tuning using the MapReduce framework.

Metrics Evaluation

The design of scalable learning systems has enabled us to address larger problems easily. Algorithms employing MapReduce or RDD may take less training time and may result in an increase of speed on a huge dataset compared to its serial counterpart. Designing a big data algorithm entails two main challenges, which are also pointed out by Cordeiro et al.[35] in his research. The two main challenges are to minimize the I/O cost and network communication cost. Researchers have employed various ways to overcome these challenges and to achieve high performance. The parallel data-intensive machine algorithm covered in previous section could be thus compared on various evaluation metrics. The evaluation metrics could be divided in two categories, namely, optimization metrics and performance metrics. The brief description of each is as follows:

Optimization Metrics

Optimization metrics refers to the measures that can be used to improve the overall performance by reducing I/O cost or network communication cost or both. They help in achieving objective such as load balancing, resource utilization, and dimensionality reduction. The various optimization criteria that could be employed by distributed big data algorithms are:-

(**a**) Sampling: Sampling refers to selecting a subset of data from the entire dataset. This helps to achieve similar accuracy with reduced network communication costs and processing time. Sampling also helps to handle data skewness.

(**b**) Indexing: Indexing data may help in selecting specific data or removing unnecessary data from processing, resulting in reduced I/O cost.

(**c**) Intelligent partitioning: Data among different nodes should not be partitioned randomly as it may lead to increased data shuffling among nodes. In order to reduce the communication cost, an intelligent data partitioning strategy may be adopted. Mappers distribute the output to reducers using hashing. Data skewness can happen at a reducer when one reducer is heavily loaded or is taking more time to compute compared to others. A new job cannot start until all reducers have finished, resulting in the improper utilization of resources. This can be reduced by exploiting data locality and using a load-aware partitioning strategy, instead of blind hashing for distribution among reducers.[59,60] A dynamic data partitioning method can also continuously optimize the varying size of data for each iteration, providing load balancing.[34] Thus, an optimal partitioning may help in handling imbalanced data skew and load balancing.

(d) Special Data Structure: Big data implies a huge dataset. Learning or overall computation of a machine-learning algorithm for big data may require multiple MR jobs, resulting in increased I/O costs. It may also be required to share information among nodes, entailing a high communication cost. Designing a special data structure to hold specific information may reduce the computational cost and would help in some form of data caching or data aggregation.

(e) In-memory computation: In-memory computation helps to cache intermediate data in main memory and perform computation. This reduces the number of disk I/O required after every step of a typical MR job.

(f) Asynchronous execution: A typical MR job is synchronous in nature, which implies that all the reducers must finish before a new map could be started. This results in the ineffective utilization of resources.

Papers were evaluated for the abovementioned optimization metrics, as presented in Table 2. Papers are assigned a value 1 in case the metric is used and a value 0 in case the metric is not used in the algorithm.

Performance Metrics

Performance metrics are used to compare the performance of the algorithm against its serial counterpart. Papers were evaluated on the following mentioned performance metrics and is presented in Table 2.

(a) Data source validation: The data are used to validate the algorithm. A synthetic dataset may result in poor validation but may help in case no real dataset is available or to test for specific test cases. The values assigned are as follows:

Data Source	Value Assigned
Synthetic datasets	0
Real-world datasets	1
Both (Synthetic datasets + Real-world datasets)	2

(b) Comparative analysis: It is used to check whether the results obtained have been compared with other techniques. The values assigned are as follows:

Comparative Analysis	Value Assigned
No comparison	0
Comparison with the base technique or similar serial technique	1
Comparison among parallel techniques (MapReduce vs Spark)	2

(c) Number of MR jobs: It is a measure of the number of MR jobs required to complete the algorithm. The number of MR jobs determines the amount of I/O cost as every time data needs to be written to disk and read from disk. More number of MR jobs may also result in increased network communication costs. Spark would help to achieve better performance in case of multiple MR jobs due to in-memory computation. The values assigned are as follows:

No. of MR Jobs	Value Assigned
Low	0
High	1

(d) Speedup: Speedup determines the reduction in time to compute or train by parallel algorithm as compared to its serial counterpart. Parallelism may result in increased speed, which may not be always linear due to high network communication costs among nodes. The values assigned are as follows:

Speedup	Value Assigned
Good speedup due to MapReduce	0
Excellent speedup in case of MapReduce due to optimizations metric employed	1
Excellent speedup due to in-memory computation such as Spark	2

(e) Accuracy: Accuracy determines the quality of data output as compared to its serial counterpart. The parallel and distributed big data algorithms may result in the decrease of output accuracy due to data distribution. A balance may be needed between output accuracy and speed. The values assigned are as follows:

Data Source	Value Assigned
Not measured	0
Approximate match	1
Perfect match	2

(**f**) Scalability: MapReduce or RDD have the basic aim of providing scalability, which means that the system would be able to gracefully handle the growth in data or number of nodes. Accuracy may be reduced for small datasets or for a large number of data nodes due to network overhead and data distribution. Thus, an optimal number of data nodes may be required. The values assigned are as follows:

Scalability	Value Assigned
Almost linear scale-up with increase in the number of nodes and datasets	0
Almost linear scale-up with a consideration of imbalanced/skewed data	1
System is scalable and detects lower and/or upper bound for parallelism degree	2

A comparative analysis of the parallel data-intensive machine-learning algorithm based on the above metrics is presented below in Table 2. There are various machine-learning toolkits such as MLib, Mahout, scikiy-learn, MLpack, Flink, and Amazon Machine Learning available for various big data computational frameworks.[61] These toolkits implement some of the machine-learning algorithms that could also be initially leveraged by the researchers for conducting their research.

DISCUSSION

MapReduce or RDD have been adopted by many researchers due to its flexibility, scalability, and easiness in handling data-intensive tasks. They help in achieving data parallelism on a large volume of data and may result in good speedup.

A number of scalable machine-learning algorithms for big data have been discussed in this article. In general, a big data-learning algorithm, as prevalent from the literature review, executes a serial algorithm for local computation at various nodes/partitions in parallel, with the help of mappers, the results of which are aggregated by the reducer/s. This task is often known as an MR job. Depending on the learning technique or fine tuning required. a single or multiple MR jobs can be executed before achieving a global view. The goal of all the abovementioned algorithms is to make a serial algorithm scalable by adapting it to run parallel on distributed platforms.

The various algorithms presented take less time than their serial counterparts. However, choosing the computational framework to employ may be decided by the context of problems. MapReduce is the best fit where the number of MR jobs is low, i.e., 1 or 2. However, in case of an iterative task, a large number of MR jobs may be required, for which MapReduce may not be the best computational framework. It has been found that for such cases, the Spark framework helps in achieving excellent speedup and better scalability due to in-memory computation and less overhead in setting up jobs for every iteration. It is further found that Spark performs better than MapReduce in almost all the cases due to its in-memory computation, which reduces the network I/O cost. Nonetheless, Spark performs better until the time it has enough memory to cache data.

This paper has also evaluated the various algorithms on optimization metrics such as sampling, intelligent partitioning, indexing, and special user-defined data structure that reduces the number of MR jobs or data shuffling, resulting in lower I/O costs and better speedup. Another factor of importance is imbalanced or skewed datasets, which may cause uneven loads on the data nodes. The consideration of imbalanced data results in load balancing and improved scalability.

The various algorithms are further evaluated on various performance metrics. It was found that the accuracy of parallel data-intensive algorithms may be reduced due to small datasets or a large number of data nodes. The advantage of parallelism may be diminished due to network and data distribution overhead. A balance is needed between speed and accuracy, which could be achieved by selecting an optimal number of data nodes.

The power and value of big data analytics provides a wealth of information. However, while designing or selecting a scalable machine-learning algorithms for big data, measures should be taken to reduce the I/O and network communication cost. Another issue that should also be considered is the handling of false positives or output validation.

TABLE 2 | Comparative Analysis of Scalable Machine-Learning Algorithms for Big Data

				Optimization Metrics						Performance Metrics					
Machine Learning Class	Paper	Adapted From	Framework Employed	Sampling	Indexing	Intelligent Partitioning	Special Data Structure	In-Memory Computation	Asynchronous Execution	Data Source Validation	Comparative Analysis	No. of MR Jobs	Speedup	Accuracy	Scalability
Clustering	Parallel K-means[25]	K-means	MapReduce	0	0	0	0	0	0	0	0	1	0	0	0
	K-means‖[28]	K-means++	MapReduce	1	0	0	0	0	0	1	1	1	1	1	0
	MR-DBSCAN[29]	DBSCAN	MapReduce	0	1	1	0	0	0	1	1	0	1	0	1
	Cludoop[30]	CluC	MapReduce	0	1	0	0	0	0	1	1	0	1	1	0
	SHAS[31]	SHC (single-linkage hierarchical clustering)	Spark	0	0	0	1	1	1	0	2	1	2	0	0
	MR-PIC[33]	pPIC	MapReduce	0	0	0	0	0	0	0	1	0	0	1	0
	CLUS[34]	SUB-CLU (subspace clustering)	Spark	0	1	1	0	1	1	2	2	1	2	2	0
	BoW:ParC and SnI[35]	Subspace clustering	MapReduce	1	0	1	0	0	0	2	1	0	1	1	0
	DisCo[36]	Coclustering	MapReduce	0	0	0	0	0	0	1	0	1	1	0	2
Classification	MRC4.5[37]	C4.5	MapReduce	0	0	1	1	0	0	0	1	1	1	0	0
	MR-tree[38]	ID3	MapReduce	0	0	0	0	0	0	1	0	1	0	0	0
	CurboSpark[39]	CourboTree—a time series decision tree	Spark	0	0	0	0	1	1	0	0	1	2	0	0
	SMRF[40]	Random forest	MapReduce	0	0	0	0	0	0	1	1	0	0	2	0
	PLANET[41]		MapReduce	0	0	0	1	0	0	1	0	0	1	1	2
	MreC4.5[42]	Ensemble C4.5	MapReduce	0	0	0	0	0	0	1	0	0	0	0	0
	Chi-FRBCS-BigData[44]	Chi-FRBCS (Fuzzy rule-based classification systems)	MapReduce	0	0	0	0	0	0	1	1	0	0	1	2

(*continued overleaf*)

TABLE 2 | Continued

				Optimization Metrics						Performance Metrics					
Machine Learning Class	Paper	Adapted From	Framework Employed	Sampling	Indexing	Intelligent Partitioning	Special Data Structure	In-Memory Computation	Asynchronous Execution	Data Source Validation	Comparative Analysis	No. of MR Jobs	Speedup	Accuracy	Scalability
	MR version of naive Bayes with rough set[45]	Naive Bayes with rough set	MapReduce	0	0	0	0	0	0	1	1	0	0	1	0
	MR version of naive Bayes Classifier[46]	Naive Bayes classifier	MapReduce	0	0	0	0	0	0	1		0	0	1	2
	MapReduce parallel SVM[47]	LibSVM	Twister	0	0	0	0	1	0	1	0	1	0	1	2
	PSMR[48]	LibSVM	MapReduce	0	0	0	0	0	0	1	1	1	0	1	0
	MRSVM[49]	LibSVM	MapReduce	0	0	0	0	0	0	1	1	1	0	1	0
	PBPNN[50]	BPNN	MapReduce, Haloop, and Spark	1	0	0	0	1	1	1	2	0	2	1	0
	MRBP[51]	BPNN	MapReduce	0	0	0	0	0	0	1	1	0	0	1	0
	MLR–MR[52]	MLR	MapReduce	0	0	0	0	0	0	2	1	0	0	1	0
Semi-supervised learning	Multiview learning on MapReduce using cotraining[53]	Cotraining	MapReduce	0	0	0	1	0	0	2	2	0	1	0	0
Deep learning	Large-scale deep belief nets with MapReduce[58]	DBN with RBM	MapReduce	0	0	0	0	0	0	1	1	1	0	1	0

BPNN, back-propagation neural network; PBPNN, parallel BPNN; DBN, deep belief network; MLR, multiple linear regression; MLR–MR, multiple linear regression on MapReduce; SVM, support vector machine; PSMR, parallel SVM on MapReduce; MRSVM, MapReduce SVM; MRBP, MapReduce back propagation; RBM, restricted Boltzman machine.

LIMITATIONS

The purpose of this study is to provide a brief overview of big data analytics. This study has provided a comprehensive literature review of data-intensive parallel machine-learning algorithms for big data analytics. A number of important limitations of the study are:

(**a**) The study has not considered parallel algorithms employing vertical scaling, such as GPUs, due to their limited scalability. The study has also not addressed parallel algorithms employing MPI.

(**b**) The study has not covered graph-based analytical algorithms as they best suit the BSP (Bulk Synchronous Parallel) model rather than MapReduce or RDD.

(**c**) The study has provided just a brief overview of various techniques and algorithms. However, in order to see the full details regarding particular techniques or algorithms, the corresponding paper may be referred.

(**d**) The study is limited to only data-intensive batch algorithms and does not consider online streaming algorithms.

RESEARCH IMPLICATIONS

Big data analytics is evolving in nature, and its potential has been exploited in almost every area and industry, such as telecom, finance, medicine, and government agencies, to produce better outcomes, leading to improved growth and success. Big data analytics has been found to be significant in various areas such as national development, industrial upgrade, scientific research, interdisciplinary research, and better prediction.[62] It has been used to solve complex problems such as recommendation, content filtering, fraud detection, and terrorist tracking. A typical example of big data analytics demonstrating its power is an image recognition system, which took only 5 min to study 60,000 handwritten digits on a Spark cluster running multiple deep learning jobs.[63]

This research would help the researchers working in the area of big data analytics to choose the right computational framework based on their underlying context. This research would help other researchers in understanding which machine-learning techniques have been adapted to MapReduce, making them scalable in prior studies. This would also help researchers in developing a new framework or hybrid model for big data analytics. Furthermore, the various evaluation metrics could be used as a benchmark in evaluating any new technique for big data.

CONCLUSION AND FUTURE WORK

The paper presents an overview of big data analytics, with a special focus on data-intensive distributed machine-learning algorithms for big data. The present paper also performs a comparative analysis of the parallel data-intensive machine-learning algorithm on optimization and performance metrics. The design of scalable and fault tolerant learning systems have enabled us to address larger problems easily due to which big data analytics has drawn attention of many researchers. This study would provide a base for the researchers in this area as it provides a wide collection of previous research.

This review is theoretical in nature. However, in the future, we would also like to implement scalable machine-learning algorithms on real-world problems, such as log analysis and social media analysis. Based on our survey, most of the previous studies have focused on making traditional serial techniques scalable. A new technique built from scratch for big data is missing in the literature, which could be considered in future work. Designing a hybrid technique based on MapReduce and MPI, utilizing the merits of both, is also worth researching in future. Visualization plays an important role in big data analytics and has not been explored and is, therefore, also an important topic of exploration.

REFERENCES

1. Owais SS, Hussein NS. Extract five categories CPIVW from the 9V characteristics of the big data. *Int J Adv Comput Sci Appl* 2016, 7:254–258.
2. Wu X, Zhu X, Wu GQ, Ding W. Data mining with big data. *IEEE Trans Knowl Data Eng* 2014, 26:97–107.
3. Chen H, Chiang RHL, Storey VC. Business intelligence and analytics: from big data to big impact. *MIS Q* 2012, 36:1165–1188.
4. Singh D, Reddy CK. A survey on platforms for big data analytics. *J Big Data* 2014, 2:1–20. doi:10.1186/s40537.014.0008.6.

5. Tsai C, Lai C, Chao H, Vasilakos AV. Big data analytics: a survey. *J Big Data* 2015, 2:1–32. doi:10.1186/s40537.015.0030.3.

6. Nathalie J, Stefanowski J. Big data analysis: new algorithms for a new society. In: *A Machine Learning Perspective on Big Data Analysis*. Springer International Publishing: Cham; 2016, 1–31.

7. Cohen J, Dolan B, Dunlap M, Hellerstein J, Welton C. Mad skills: new analysis practices for big data. *Proc VLDB Endow* 2009, 2:1481–1492. doi:10.14778/1687553.1687576.

8. Schmarzo B. Understanding the role of Hadoop in your bi environment. Available at: https://infocus.emc.com/williamschmarzo/understandingtheroleofhadoop-inyourbienvironment/. (Accessed February 6, 2016).

9. Borthakur D. HDFS architecture guide. Available at: https://hadoop.apache.org/docs/r1.2.1/hdfsdesign.pdf. (Accessed February 6, 2016).

10. Bakshi K. Considerations for big data: architecture and approach. In: *Proceedings of the 2012 I.E. Aerospace Conference*, Big Sky, Montana, US, Mar 3–10, 2012, 1–7. doi:10.1109/AERO.2012.6187357.

11. Kulkarni AP, Khandewal M. Survey on hadoop and introduction to YARN. *Int J Emerging Technol Adv Eng* 2014, 4:82–87.

12. Roman J. Hadoopecosystemtable. Available at: https://hadoopecosystemtable.github.io. (Accessed February 6, 2016).

13. Brewer E. CAP twelve years later: how the "rules" have changed. *Computer* 2012, 45:23–29. doi:10.1109/MC.2012.37.

14. Sharma S, Tim US, Gadia SK, Wong JS, Shandilya R, Peddoju SK. Classification and comparison of NoSQL big data models. *Int J Big Data Intell* 2015, 2:201–221.

15. Moniruzzaman ABM, Hossain SA. NoSQL database: new era of databases for big data analytics—classification, characteristics and comparison. *Int J Database Theor Appl* 2013, 6:1–14.

16. Moniruzzaman ABM. NewSQL: towards next-generation scalable RDBMS for online transaction processing (OLTP) for big data management. International Journal of Database Theory & Application 2014, 7:121–130.

17. Sundaresan B, Kandavel D. Big languages for big data: a study and comparison of current trend in data processing techniques for big data. In: *Design and Implementation of Programming Languages Seminar*, TU, Darmstadt, Germany, 2014. doi: 10.13140/2.1.5107.8402.

18. Lin J. Mapreduce is good enough? If all you have is a hammer, throw away everything that's not a nail! Big Data 2012, 1:28–37.

19. Doulkeridis C, Nørvåg K. A survey of large-scale analytical query processing in MapReduce. *VLDB J* 2013, 23:355–380. doi:10.1007/s00778.013.0319.9.

20. Landset S, Khoshgoftaar T, Richter A, Hasanin T. A survey of open source tools for machine learning with big data in the Hadoop ecosystem. *J Big Data* 2015, 2:1–36. doi:10.1186/s40537.015.0032.1.

21. Zaharia M, Chowdhury M, Das T, Dave A, Ma JM, McCauley M, Franklin MJ, Shenker S, Stoica I. Fast and interactive analytics over hadoop data with Spark. *USENIX* 2012, 37:45–51.

22. Shahrivari S, Jalili S. Beyond batch processing: towards real-time and streaming big data. *Computers* 2014, 3:117–129.

23. Marz N, Warren J. *Big Data: Principles and Best Practices of Scalable Realtime Data Systems*. Greenwich, CT: Manning Publications Co; 2015.

24. Vizard M. Cloudera aims to replace MapReduce with Spark as default hadoop framework. Available at: http://www.datacenterknowledge.com/archives/2015/09/09/cloudera-aims-to-replace-mapr. (Accessed February 6, 2016).

25. Zhao W, Ma H, He Q. Parallel K-means clustering based on MapReduce. In: *Proceedings of First International Conference of Cloud Computing (CloudCom)*, Springer, Beijing, China, 2009, 674–679.

26. Thomas L, Annappa B. Application of parallel K-means clustering algorithm for prediction of optimal path in self aware mobile ad-hoc networks with link stability. In: *Proceedings of First International Conference of Advances in Computing and Communications*, Springer, Kochi, India, July 22–24, 2011, 396–405.

27. Arthur D, Vassilvitskii S. K-means++: the advantages of careful seeding. In: *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (SODA '07), Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2007, 1027–1035.

28. Bahmani B, Moseley B, Vattani A, Kumar R, Vassilvitskii S. Scalable k-means++. *PVLDB* 2012, 5:622–633.

29. He Y, Tan H, Luo W, Feng S, Fan J. MR-DBSCAN: a scalable MapReduce-based DBSCAN algorithm for heavily skewed data. *Front Comput Sci* 2014, 8:83–99. doi:10.1007/s11704.013.3158.3.

30. Yu Y, Zhao J, Wang X, Wang Q, Zhang Y. Cludoop: an efficient distributed density-based clustering for big data using Hadoop. *Int J Distr Sensor Network* 2015, 2015:1–13. doi:10.1155/2015/579391.

31. Jin C, Liu R, Chen Z, Hendrix W, Agrawal A, Choudhary A. A scalable hierarchical clustering algorithm using Spark. In: *Proceedings of IEEE First International Conference on Big Data Computing Service and Applications (BigDataService)*, San Francisco Bay, CA, USA, 2015, 418–426. doi: 10.1109/BigDataService.2015.67.

32. Yan W, Brahmakshatriya U, Xue Y, Gilder M, Wise B. p-pic: parallel power iteration clustering for big data. *J Parallel Distr Comput* 2013, 73:352–359.

33. Jayalatchumy D, Thambidurai P. Implementation of p-pic algorithm in MapReduce to handle big data. *Int J Res Eng Technol* 2014, 3:113–118.

34. Zhu B, Mara A, Mozo A. CLUS: parallel subspace clustering algorithm on spark. In: *Proceedings of New Trends in Databases and Information Systems*, Poitiers, France, 2015, 175–185.

35. Cordeiro RLF, Traina JC, Traina AJM, Lo´pez J, Kang U, Faloutsos C. Clustering very large multi-dimensional datasets with MapReduce. In: *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '11*, ACM: New York, NY, USA, 2011, 690–698. doi: 10.1145/2020408.2020516.

36. Papadimitriou S, Sun J. Disco: distributed co-clustering with map-reduce: a case study towards petabyte-scale end-to-end mining. In: *Proceedings of the Eighth IEEE International Conference on Data Mining, ICDM '08*, IEEE Computer Society: Washington, DC, USA, 2008, 512–521, doi: 10.1109/ICDM.2008.142.

37. Dai W, Ji W. A MapReduce implementation of C4. 5 decision tree algorithm. *Int J Database Theor Appl* 2014, 7:49–60.

38. Purdila V, Pentiuc S. MR-tree—a scalable MapReduce algorithm for building decision trees. *J Appl Computer Sci Math* 2014, 16:16–19.

39. Salperwyck C, Maby S, Cubillé J, Lagacherie M. CourboSpark: decision tree for time-series on Spark. In: *Proceedings of the 1st International Workshop on Advanced Analytics and Learning on Temporal Data*, Porto, Portugal, 2015.

40. Han J, Liu Y, Sun X. A scalable random forest algorithm based on MapReduce. In: *Proceedings of the 4th IEEE International Conference on Software Engineering and Service Science* (*ICSESS*), Beijing, China, May 23–25, 2013, 849–852. doi: 10.1109/ICSESS.2013.6615438.

41. Panda B, Herbach J, Basu S, Bayardo RJ. Planet: massively parallel learning of tree ensembles with MapReduce. *PVLDB* 2009, 2:1426–1437.

42. Wu G, Li H, Hu X, Bi Y, Zhang J, Wu X. MReC4.5: C4.5 ensemble classification with MapReduce. In: *Proceedings of the Fourth ChinaGrid Annual Conference*, Yantai, China, Aug 21–22, 2009, 249–255. doi: 10.1109/ChinaGrid.2009.39.

43. Singh K, Guntuku SC, Thakur A, Hota C. Big data analytics framework for peer-to-peer botnet detection using random forests. *Inform Sci* 2014, 278:488–497. doi:10.1016/j.ins.2014.03.066.

44. Río SD, López V, Benítez JM, Herrera F. A MapReduce approach to address big data classification problems based on the fusion of linguistic fuzzy rules. *Int J Comput Intell Syst* 2015, 8:422–437. doi:10.1080/18756891.2015.1017377.

45. Dai Y, Sun H. The naive Bayes text classification algorithm based on rough set in the cloud platform. *J Chem Pharmaceut Res* 2014, 6:1636–1643.

46. Liu B, Blasch E, Chen Y, Shen D, Chen G. Scalable sentiment classification for big data analysis using naive Bayes classifier. In: *Proceedings of IEEE International Conference on Big Data*, Silicon Valley, CA, USA, 2013, 99–104.

47. Sun ZQ, Fox GC. Study on parallel SVM based on MapReduce. In: *International Conference on Parallel and Distributed Processing Techniques and Applications*, Las Vegas, NV, USA, 2012, 495–561.

48. Xu K, Wen C, Yuan Q, He X, Tie J. A MapReduce based parallel SVM for email classification. *J Network* 2014, 9:1640–1647.

49. Khairnar J, Kinikar M. Sentiment analysis based mining and summarizing using SVM- Mapreduce. *Int J Comput Sci Inform Tech* 2014, 5:4081–4085.

50. Liu Y, Xu L, Li M. The parallelization of back propagation neural network in MapReduce and Spark. *Int J Parallel Program* 2016, 1–20. doi:10.1007/s10766.016.0401.1.

51. Zhang H, Xiao N. Parallel implementation of multilayered neural networks based on MapReduce on cloud computing clusters. *Soft Comput* 2015, 20:1471–1483. doi:10.1007/s00500.015.1599.3.

52. Rehab MA, Boufarès F. Scalable massively parallel learning of multiple linear regression algorithm with MapReduce. In: *IEEE International Conference Trustcom/BigDataSE/ISPA*, Helsinki, Finland, Aug 20–22, 2015, 41–47. doi:10.1109/Trustcom.2015.560.

53. Hariharan C, Shivashankar S. Large scale multi-view learning on MapReduce. In: *Proceedings of 19th International Conference on Advanced Computing and Communications*, Chennai, India, 2013.

54. Zhao Q, Ye F, Wang S. A new back-propagation neural network algorithm for a big data environment based on punishing characterized active learning strategy. *Int J Knowl Syst Sci* 2013, 4:32–45. doi:10.4018/ijkss.2013100103.

55. Li Y, Schuurmans D. Mapreduce for parallel reinforcement learning. In: *Proceedings of the 9th European Conference on Recent Advances in Reinforcement Learning*, EWRL'11, Springer-Verlag: Berlin, Heidelberg, Germany, 2012, 309–320.

56. Barrett E, Duggan J, Howley E. A parallel framework for Bayesian reinforcement learning. *Connection Sci* 2014, 26:7–23. doi:10.1080/09540091.2014.885268.

57. Chen XW, Lin X. Big data deep learning: challenges and perspectives. *IEEE Access* 2014, 2:514–525. doi:10.1109/ACCESS.2014.2325029.

58. Zhang K, Chen XW. Large-scale deep belief nets with MapReduce. *IEEE Access* 2014, 2:395–403. doi:10.1109/ACCESS.2014.2319813.
59. Ibrahim S, Jin H, Lu L, He B, Antoniu G, Wu S. Handling partitioning skew in MapRe duce using LEEN. *Peer-to-Peer Network Appl* 2013, 6:409–424. doi:10.1007/s12083.013.0213.7.
60. Chen Q, Yao J, Xiao Z. Libra: lightweight data skew mitigation in MapReduce. *IEEE Trans Parallel Distr Syst* 2015, 26:2520–2533. doi:10.1109/TPDS.2014.2350972.
61. Desale D. Top 15 frameworks for machine learning experts. Available at: http://www.kdnuggets.com/2016/04/top-15-frameworks-machine-learning-experts.html. (Accessed June 30, 2016).
62. Xiaolong J, Benjamin WW, Xueqi C, Yuanzhuo W. Significance and challenges of big data research. *Big Data Res* 2015, 2:59–64. doi:10.1016/j.bdr.2015.01.006.
63. Riggins J. Apache spark for deep learning: two case studies. Available at: http://thenewstack.io/two-tales-big-data-framework-adoption-apache-spark/. (Accessed June 30, 2016).