




Review

A Comprehensive Survey of MapReduce Models for Processing Big Data

Hemn Barzan Abdalla ^{1,*} , Yulia Kumar ² , Yue Zhao ¹ and Davide Tosi ³ ¹ Department of Computer Science, Wenzhou-Kean University, Wenzhou 325015, China; yuezhao@kean.edu² Department of Computer Science and Technology, Kean University, Union, NJ 07083, USA; ykumar@kean.edu³ Department of Theoretical and Applied Sciences, University of Insubria, 21100 Varese, Italy; davide.tosi@uninsubria.it

* Correspondence: habdalla@kean.edu

Abstract: With the rapid increase in the amount of big data, traditional software tools are facing complexity in tackling big data, which is a huge concern in the research industry. In addition, the management and processing of big data have become more difficult, thus increasing security threats. Various fields encountered issues in fully making use of these large-scale data with supported decision-making. Data mining methods have been tremendously improved to identify patterns for sorting a larger set of data. MapReduce models provide greater advantages for in-depth data evaluation and can be compatible with various applications. This survey analyses the various map-reducing models utilized for big data processing, the techniques harnessed in the reviewed literature, and the challenges. Furthermore, this survey reviews the major advancements of diverse types of map-reduce models, namely Hadoop, Hive, Pig, MongoDB, Spark, and Cassandra. Besides the reliable map-reducing approaches, this survey also examined various metrics utilized for computing the performance of big data processing among the applications. More specifically, this review summarizes the background of MapReduce and its terminologies, types, different techniques, and applications to advance the MapReduce framework for big data processing. This study provides good insights for conducting more experiments in the field of processing and managing big data.

Keywords: MapReduce; data processing; big data; Hadoop; data mining



Academic Editor: Carson K. Leung

Received: 11 February 2025

Revised: 18 March 2025

Accepted: 20 March 2025

Published: 27 March 2025

Citation: Abdalla, H.B.; Kumar, Y.; Zhao, Y.; Tosi, D. A Comprehensive Survey of MapReduce Models for Processing Big Data. *Big Data Cogn. Comput.* **2025**, *9*, 77. <https://doi.org/10.3390/bdcc9040077>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The rapid development of information technologies has led the way to the rise of the five qualities of big data: veracity, volume, variety, value, and velocity. The conventional methods employed for processing the data face challenges in handling complicated and vast amounts of data [1]. The major tasks are data classification and feature selection, and numerous algorithms have been developed for processing both tasks. For the effective categorization of big data, the selection of a suitable approach for specific tasks is necessary [2], and for data mining processes, machine learning (ML) algorithms are considered effective. ML algorithms exhibit better performance for even a large range of big data [3]. Learning algorithms, including decision trees and logistic regression, cannot often handle imbalance problems. For larger datasets, classifiers, namely support vector machines (SVMs) and extreme learning machines (ELMs), can exhibit higher classification performance [4]. However, the ELM classifier also experiences hardships from imbalance issues, but the weighted ELM can provide accurate classification by selecting the weights for the classifier model [5]. When the processing is performed on a single system, the complexity is higher

for managing and storing large amounts of data. Therefore, the development of classifiers that can effectively process large amounts of data is necessary [6].

Several works have been conducted to explore the parallel processing approaches that can implement classification algorithms without affecting the results [7]. However, longer processing times make it difficult to analyze the sequence of classes [8]. In each class of the database, the existing samples are different, and, in most cases, the difference in the available samples is massive [9]. In contrast with the sample size of the dataset, the number of characteristics is limited. Hence, the need for a technique to process the data without compromising the features has increased [10]. MapReduce is one of the most effective parallel processing systems that can be harnessed to improve the processing speed of any task. Although MapReduce has been developed for Hadoop architecture, the method is utilized in various implementation approaches due to the wide range of utilities [11]. Map reducing effectively performs in a distributed manner simultaneously over large datasets. Map-reduce systems possess the ability to be harmonious with various applications to perform in-depth evaluation of the data [12]. The map-reduce models' performance depends on the system's configuration within the node clusters. The MapReduce model for processing large-scale datasets depends on the Hadoop Distributed File System (HDFS) for storage, which is characterized by its scalability, fault tolerance, and data locality.

The configuration decides whether the node is multi- or single-node and controls the parameters of the Hadoop network. This defines the split size of processing parameters and reduces it for task distribution [13]. Map-reducing utilizes polling data to enhance cluster classification. A wide range of data is divided into smaller segments in the Hadoop network by the head node in the cluster, and the information is circulated within the slave nodes [14]. Each node in the cluster comprises a map-reducing structure, and the mapper class processes the information in the cluster. The processed information is collected by a separate class and concatenated to produce the outcome [15]. Apache Flume can be utilized for the extraction of real-time information, and Apache Hive structures the information stored in the Hadoop Distributed File System (HDFS) [16]. The highly complex information can be structured by the Hive utilizing the Structured Query Language (SQL) known as Hive Query Language (HiveQL) [17]. Further, the most challenging problem in the selection of an effective job scheduling strategy is to guarantee the Service-Level Agreement (SLA) associated with the scheduled tasks. More specifically, the SLA metrics, including the response time, throughput, error rate, latency, capacity, and resolution time, indicate the service quality and availability, and SLA violation results in increased costs, including reputational damage, and potentially higher power consumption caused by inefficient resource utilization. The system's response time and data transaction speed can affect the framework's scalability. Hence, the limitations, including the scalability, batch processing, SLA violation, and complexity of the parallel processing, result in the need to develop MapReduce models for big data processing [18].

The significant contribution of the proposed review is explained as follows:

- To attain a deep knowledge of map-reduce-based big data processing, this survey aims to interpret various map-reducing models focused on the methodology, the performance metrics, their advantages, utilized datasets, and limitations.
- Around 75 articles related to MapReduce for processing big data are reviewed in this research, which provides good insight into choosing the appropriate model for processing various data, the challenges faced, and future directions.
- More specifically, this review describes different types of map-reduce models, including Hadoop, Hive, Spark, Pig, MongoDB, and Cassandra.
- In addition, this review analyzed various metrics for evaluating MapReduce frameworks' performance for big data processing.

- The overview of the research benefits the researchers in overcoming the limitations in the existing approaches to develop more effective and sustainable models in the future.

The research is sorted into the following sections. Section 2 demonstrates the background of the map-reduce model. Section 3 covers the literature and the types of map-reducing models available. Section 4 represents the analysis of the utilized datasets while the performance metrics are discussed in Section 5. Section 6 discusses the research gaps with future scopes, and the conclusion is presented in Section 7.

2. Background of MapReduce Model

MapReduce is a simplified programming language implemented for processing various datasets in intensive applications, as illustrated in Figure 1. The process of MapReduce involves three phases, namely, the shuffle, the reduce stage, and the map stage. The input data are processed from the files stored in the HDFS in the map stage and converted to pairs of values and keys. The outcome from the map stage is processed in the reduce stage into new data, which is again stored in the HDFS. The schedules and the execution plans are coordinated by the job tracker and task tracker, which categorizes tasks into maps and reduces functions. The report status and execution map are allocated for each task record to maintain the report of the processed tasks. Various systems are implemented with the MapReduce model, such as Yet Another Resource Negotiator (YARN), Hadoop, Amazon EC2, Spark, Hive, Pig, Cassandra, and MongoDB. YARN is a resource manager that utilizes the MapReduce method to process data efficiently. The implementation of security controls and the generation of high-availability features are executed with operational applications. Hadoop derives inspiration from the Google File System (GFS), which provides scalable infrastructure for large, distributed datasets [19]. The method processes large clusters in common servers and solves highly disputable problems. Apache Spark is highly beneficial for the graph analysis and structured processing of data. The multistage in-memory model is utilized to perform easier and distributed operations [4]. Furthermore, the upper-level libraries and Spark core manage and access diverse clusters in the Hadoop data source. Apache Hive is a fault-resilient and distributed system that allows analytics at a wider range. The Hive query language processes all the queries in the system to sizeable datasets [8]. Apache Pig allows a high level of abstraction to process large data over MapReduce, which utilizes Pig Latin to process the information [20]. The large amounts of data across the commodity servers are managed with Apache Cassandra, which provides high availability without a reduction in performance [11]. The open-source and non-relational properties of Cassandra enhance higher scalability and flexibility among distributed architectures. Figure 1 depicts the architecture of the MapReduce framework.

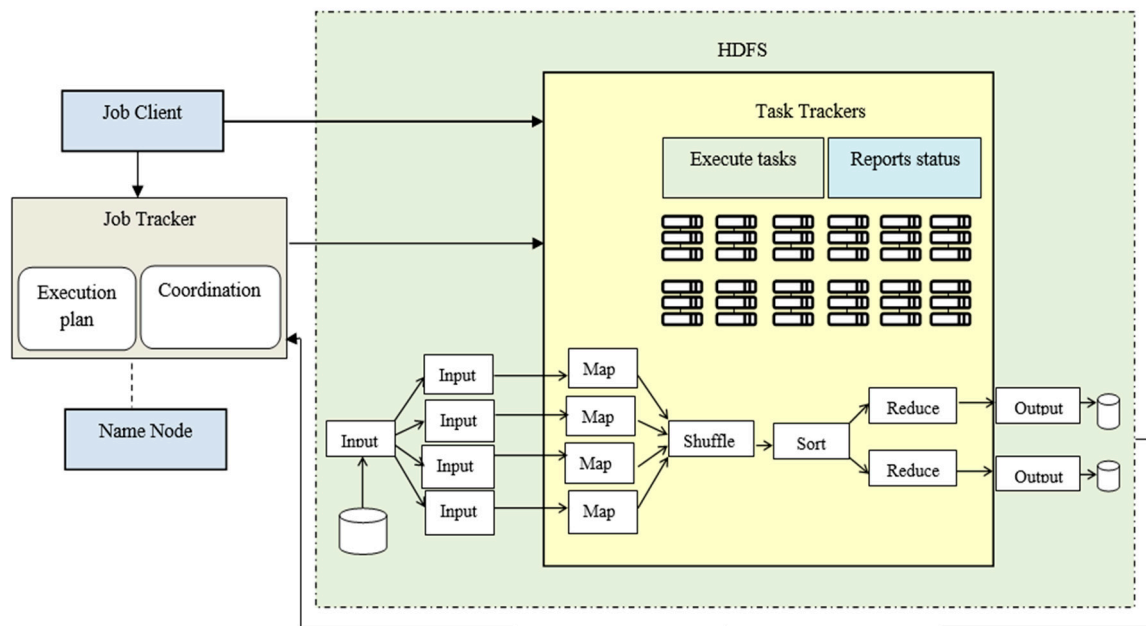


Figure 1. The architecture of the MapReduce framework.

3. Methodology

The methodology of this review focuses on the systematic approach used in selecting the articles, followed by the taxonomy of MapReduce models, and the literature review is covered in detail in the following section. A detailed literature review that gives an analysis of the MapReduce models is also included in this section.

3.1. Article Selection Process

Following the PRISMA guidelines, a systematic literature review is conducted in the proposed review to ensure a robust multi-database search strategy. Figure 2 demonstrates the steps involved in the PRISMA-based review process. Initially, the articles are filtered from the data sources using a comprehensive search. Utilizing the search strings based on the specific key terms such as “Hadoop”, “MapReduce”, and “Big Data”, this review searched for articles including the latest work available in the literature. To obtain the most recent works in the field of data mining and MapReduce, the initial search was limited to the years 2018 to 2022. As a result, the survey’s initial results gathered 712 articles that were employed to accomplish the metadata analysis. Subsequently, the process determines each article’s relevance to the research issue by evaluating the titles, abstracts, and complete texts of the filtered articles. In addition, the articles are selected according to eligibility, and the articles that are surveys, are low quality, or lack proper analysis are deleted in the selection process. Consequently, only 75 articles are filtered and applied for in-depth analysis. In order to select the journals that meet the inclusion criteria and adhere to the highest standards of research integrity, this review must follow a systematic and exacting process. This robust filtering approach obviously enhances the validity and reliability of the findings associated with this systematic review. Finding the research gaps and acquiring information on the most recent achievements and advanced mechanisms are the ultimate goals of focusing on recent publications.

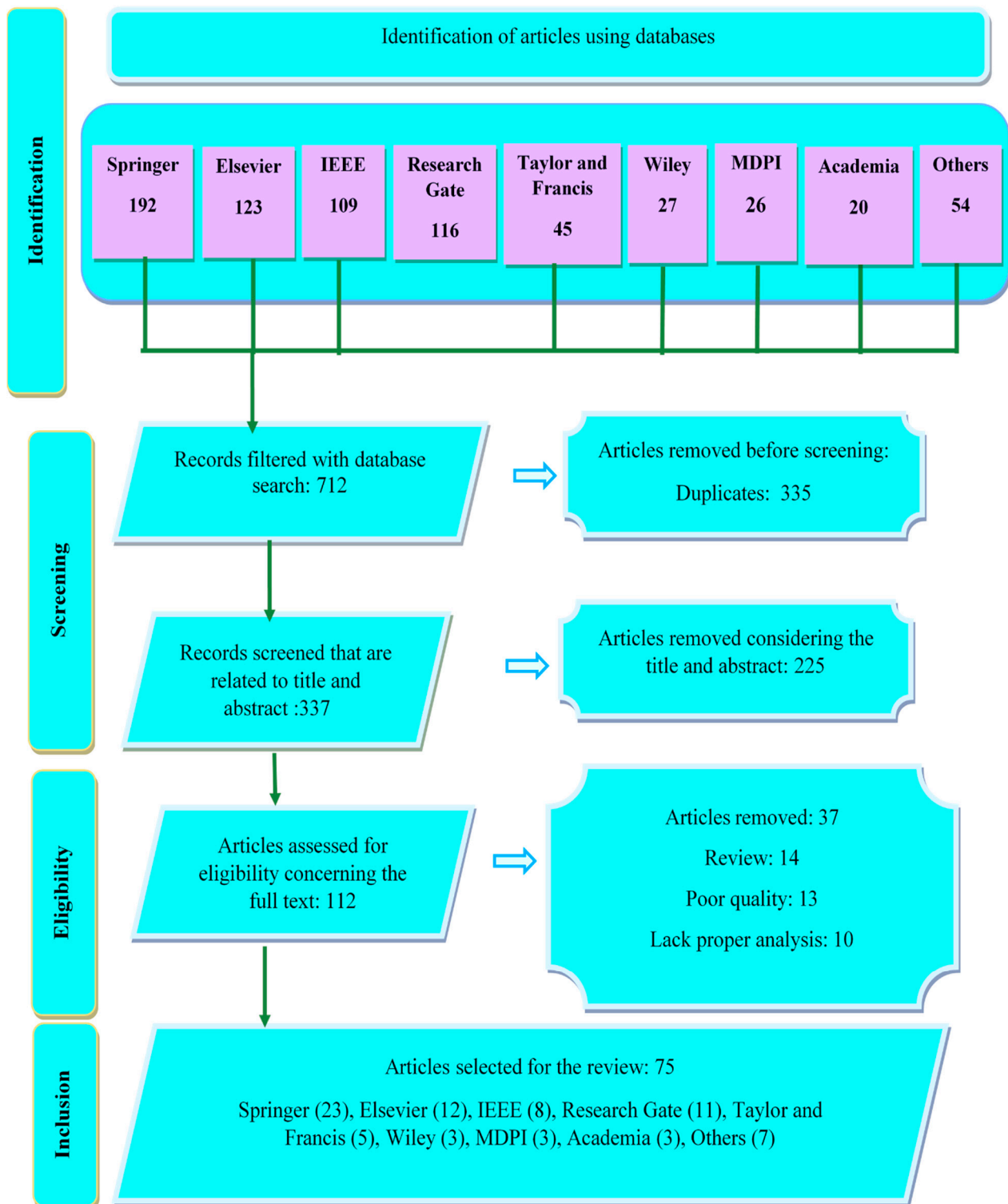


Figure 2. PRISMA flow diagram.

3.2. Literature Review

The overview of the map-reducing framework for the classification of big data utilized in the research articles is demonstrated in this section. This review includes 75 research articles, and the research questions addressed in this study are as follows:

1. What are the advantages and limitations of utilizing the MapReduce framework for big data classification?

2. Which methods are predominantly employed in the reviewed research articles, and how do they contribute to the framework's effectiveness?
3. What challenges and gaps exist in the current use of MapReduce for big data classification, and how can they be addressed?

3.3. Types of Map-Reduce Models

The sections cover the various MapReduce models utilized in the reviewed literature, such as Hadoop, Spark, Hive, Pig, Cassandra, and MongoDB. The methods utilized in each study, the advantages, achievements, and limitations are explained in this section. The taxonomy diagram of the map-reduce models is illustrated in Figure 3.

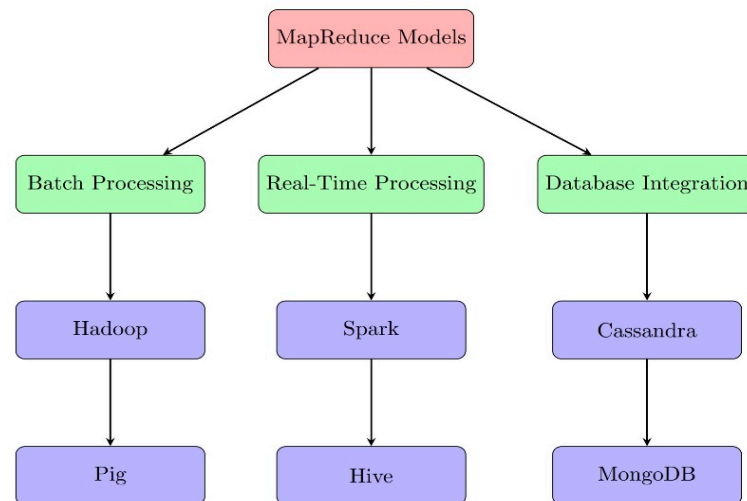


Figure 3. Taxonomy diagram of the map-reduce models.

3.3.1. Hadoop Map-Reduce Model

A. Kumar et al. [1] introduced a Hadoop map-reducing framework that revalued the output keys and value pairs, leading the way to effectively retrieving a large scale of information. The technique could inform the massive database from storage and indexing techniques for distributing large sets of queries. The data processing time of the model was lower; however, the model suffered from incompatibility between diverse nodes. D. Dahiphale [21] modeled a distributed and parallel algorithm for 2-ECCs (BiECCA), which utilized a wide range of databases that enabled continuous data processing. The algorithm utilized in the MapReduce framework managed to find the connected units in the graph, which reduced the time needed to complete the processing. The graph mining issues were solved and evaluated on a big data analytic framework; however, the small and large star processes handled only one edge at a time. N. Verma et al. [22] presented the HDFS-MapReduce-based architecture, which employed the Apriori algorithm with cloud architecture that exhibited higher performance in mining big data. They performed with low resources and computing models, reducing overhead issues during computation. The mining tool effectively retrieved crucial information from the retail transactional databases. The processing time of the HDFS-MapReduce model was lower; however, the ability to handle a wide range of information was not effective.

The Hadoop platform-based MapReduce system was introduced by S.-Y. Choi and K. Chung [3], a big data knowledge process that effectively managed the services by gathering and processing varied health information. The framework effectively processed the distributed data sources through data mining using the Hadoop map-reducing process. The associations of the relevant information were evaluated using the reliability characteristics, and the reduction in the key values improved the processing time. However, the evaluation

was not performed with unstructured information. T.H. Sardar and Z. Ansar [23] developed a MapReduce model based on Fuzzy C-Medoids that obtained efficiency in clustering big data. The developed algorithm was evaluated with various cluster sizes and documents, revealing scalable and effective performance in clustering large information. The performance of the model was more effective, and the execution time was lower with larger Hadoop clusters of different dimensions. However, the fuzzy model failed to evaluate the quality of clusters. S. Rajendran et al. [19] developed a chaotic-inspired optimization-based feature subset technique implemented with an optimal deep belief network for managing big data. The model was employed with Harris hawk optimization (HHO), which allocates the appropriate class labels to the data. HHO improved the classification performance, and the outcomes were evaluated under disparate dimensions; however, the limited quality of parameters affected the effectiveness.

S. Guan et al. [24] presented a big data secure storage method based on Hadoop architecture that dissipated the single node servers to multiple servers. The storage process was performed in both channels using distributed coordination mechanisms. The data were encrypted with the homomorphic encryption algorithm and amplified using the thread encryption node. The model's efficiency was improved; however, the computation of cipher text was ineffective.

S. Saleti and Subramanyam R.B.V. [10] developed a Co-occurrence Reverse Map (CRMAP) model that solved the sequential pattern issues regarding the big data with an updated database. Using the backward mining approach, an efficient distributed algorithm was developed to mine sequential patterns. This method employed CRMAP to manage the combinational explosion of candidate sequences, and the pruning efficiency of the model was higher with reduced processing time. However, the performance was limited to a single data processing method. T. D. Dang et al. [25] introduced a trust-based MapReduce framework that assigned trust values for maps and sensitive values for data where the values of each resource were laid on in the big data processing tasks. The maximum weighted problem minimized the overall value of all assignments subjected to various tasks. The resource utilization and trust gain were determined with the scheduling algorithm, which improved resource consumption rates but exhibited lower security while processing complex data.

G. Venkatesh and K. Arunesh [9] developed a three-layered traffic-aware clustering method that processed big data by reducing techniques. The model employed a data processing function map, and the implementation was partitioned into fixed-sized blocks in the HDFS. The data partition and aggregator reduced the network traffic by merging the traffic from multiple tasks. The response time of the model was lower with less traffic; however, the time taken for processing the data was higher. J. Jo and K.-W. Lee [26] introduced a delayed extracting–loading–transforming (DELT) model with map-reduce-based parallelization that managed the amount of big data in the loading procedures. The model utilized geospatial big data, and the data preparation time was reduced with the ETL systems. The parallelization effect of the model was higher for large amounts of data, and the optimization reduced the degradation of performance. However, the DELT model exhibited poor performance in dealing with spatial queries with geospatial big data. K. Meena and J. Sujatha [27] developed a Co-Effective and Adaptive Neuro-Fuzzy System (Co-EANFS) method for the effective handling of data. The association rule of the data was also evaluated, which managed the various levels of clustering and classification. The EANFS method improved the success ratio of the recommender system, and the prediction was performed with associative rule mining. The Hadoop model enhanced classification accuracy and prediction capabilities but relied on regression models to achieve adequate performance.

O. Kulkarni et al. [28] modeled a MapReduce framework based on Fractional Sparse fuzzy C-means (FrSparse FCM), which is based on the map and reduce function. The FCM algorithm evaluated the optimal centroids and was tuned by the reducer phase. The FCM-based method ensured the concurrent processing of big data acquired from distributed sources. The algorithm performed the clustering of big data with better progression; however, the classification accuracy of the system was lower. I. M. S. Ali and D. Hariprasad [2] presented a Hadoop map-reducing framework that minimized the imbalance issues in the classes with the utilization of hybrid feature selection. The optimized extreme learning machine (ELM) managed the imbalance issues of big data, and the black widow algorithm optimized the ELM configurations by tuning the hyperparameter values. However, the model failed to manage multiple data sources.

H. Azgomi and M. K. Sohrabi [29] developed a MapReduce-based construction of the Multiple Views Processing Plan (MR-MVPP) that solved the representation issues regarding materialized view selection. A set similarity-based join was performed on the base relation and the data with the hashing technique. The average time improvement of the model for large volumes of data was greater in real-world applications. The execution performance and coverage rate were effective; however, the responsive time taken for the analytical queries in online processing networks was higher. D.L. Chiang et al. [30] defined a Petri net model, which analyzed the flexibility of the map-reduce approach, and the real big data analysis system demonstrated the effectiveness of the PN model. The error prevention mechanisms increased the performance of the model and minimized the common errors that occurred in processing. The rule and file checker employed by Petri Net reduced the errors in the analysis system; nonetheless, there was no real-time evaluation of analytical performance. F. Abukhodair et al. [31] introduced a metaheuristic optimization-based technique on big data classification in a MapReduce (MOBDC-MR) model, extracting optimal characteristics and effectively classifying the big data. The BPOA-FS-based method reduced the computational time and improved classification accuracy. Moreover, the hyperparameter tuning enhanced the performance of the classifier; however, the processing time of the model was comparatively longer.

P. Peddi [5] presented a map-reduced Hadoop HDFS method for the efficient handling of big data. The Hadoop map-reduce model processed the unstructured data and evaluated the process of unstructured log data. The performance of the model was higher with unstructured log data; however, the model faced more replications in the block information, which reduced the accuracy. D. Kumar and V. K. Jha [17] introduced an improvised query optimization with Normalized K-means (NKM) to reduce the complexity of optimizing the queries in big data. The Secure Hash Algorithm (SHA-512) employed in the model processed the data by evaluating hash values, and the map-reduce function removed the repetition of data. The entropy calculation managed the privacy, and the NKM grouped the related information. The query optimization model exhibited higher accuracy and occupied less storage; however, the retrieval time of data was higher. A Computational Intelligence Clustering and Classification-based Big Data Visualization on MapReduce Environment (CICC-BDVMR) model was modeled by Z. Xu [32], providing effective visualization for map-reduced environments. The model utilized the Kernelized fuzzy C-means (KFCM) to cluster big data and the grasshopper optimization algorithm (GOA) to determine initial clusters. GOA effectively solved the globally constrained problems for better classification outcomes. However, detection outcomes were reduced for smaller datasets.

M. Alkathiri [33] presented a multi-dimensional geospatial data mining connected with the Apache Hadoop ecosystem, which supported the analysis of large-scale multi-spectral raster data. The spectral phase in the data was converted to a geometrical phase, allowing multiple bands to be managed simultaneously. The processing and mining per-

formance of the geospatial data was effective; however, it lacked a visualization interface. A. Gandomi et al. [34] developed a Hybrid Scheduling MapReduce priority (HybSMRP) model, which harnessed localization ID and dynamic priority techniques that increased the locality time of the model. The data locality rate achieved by the model was higher and avoided wasting resources for mapping tasks; however, the performance of the scheduler was not evaluated with larger datasets. C. Banchhor and N. Srinivasu [35] presented a Holoentropy Correlative Naive Bayes classifier and MapReduce Model (HCNB-MRM) to parallelly handle data from distributed sources. The complications in storing and analyzing big data were reduced, and the model obtained a better set of attributes from the database. The system's classification performance was improved using the probability index table, but the model was not evaluated with deep learning approaches [36].

J. Ramsingh and V. Bhuvaneswari [37] modeled MapReduce-Based Hybrid Naive Bayes Classifier-Term Frequency Inverse Document Frequency (NBC-TFIDF) that effectively extracted the diverse sentiments of people. Regarding the polarity score of the contents in online platforms, a map-reduce-based hybrid NBC is developed to categorize the sentences. The performance of the model was effective in multimode clusters; nonetheless, the map-reduced model could not exhibit higher precision. K. Alwasel et al. [16] presented a big data software-defined networking (BigDataSDNSim) framework, which was a self-contained system that resolved the issues in evaluating the solutions. BigDataSDNSim combined data processing with software-defined networking (SDN) for effective computation in real environments. The model optimized the performance of transmission and processing together for a better knowledge of the complex interactions among the components of BigDataSDNSim. The model's task scheduling was seamless. On the other hand, there was a high transmission in cloud environments. D. Xia et al. [38] introduced a Normal Distribution-based Long Short-Term Memory with a time Window system (WND-LSTM) that addressed the storage and computation issues of processing big data. WND-LSTM was a parallel processing framework that handled a large set of traffic flow data. The efficiency of the short-term traffic was improved. However, the model failed to evaluate the accuracy and scalability of flow prediction.

C. Iwendi et al. [7] introduced a temporal Louvain approach with the TF/IDF Algorithm, which effectively acquired the complex structure of the streaming data. The model categorized the documents into hierarchical structures, which helped analyze multiple complex characteristics. Although the accuracy of the model was higher, the execution time was higher for complex data. W. Chen et al. [13] modeled a map-reduce-based QoS data placement technique that reduced the system traffic, along with the communication resources of the data blocks optimized. The algorithm managed the data centers with Zipflike replica distribution. The model reduced the joint data placement issues in MapReduce applications, but the communication and energy costs were higher. A. K. Tripathi et al. [39] introduced a MapReduce-based Military Dog-based optimization (MR-MDBO) framework that handled big datasets with the optimal centroid, i.e., the framework efficiently identified the most representative data points (centroids) to improve clustering and reduce computational overhead for large-scale datasets. The clustering accuracy of the IoT-based big data showed improvements, and the speed-up analysis exhibited higher scalability; however, the model was not evaluated with real-world applications.

The two-stage map and reduce task scheduler (TMaR) was presented by N. Maleki et al. [40], which minimized the task completion time in each stage. The dynamic partition binder reduced the imbalances at each stage in heterogeneous environments. The partition binder prevented unnecessary movements between the nodes, which reduced the latency, yet the parallelism between the map and reduced tasks was not maintained. U. Narayanan et al. [41] developed a Secure Authentication and Data Sharing in Cloud

(SADS-Cloud) system that managed big data over cloud environments. The map-reduce model categorized the input file into fixed blocks and applied the encryption algorithm over each block. The model showed improvements in reducing information loss; however, it is vulnerable to issues in encryption and decryption. A map-reduce-based optimal data classification (MRODC) method was introduced by R. T. Selvi and I. Muthulakshmi [42], involving various stages of the Hadoop ecosystem. An improved K-means algorithm was integrated with the gradient boost tree, which improved the classifier results. The clustering efficiency of the MRODC model was higher, yet there was no real-time diagnosis of the patient information.

T. Chawla et al. [43] developed a Multi-Level Storage Scheme (MuSe) for solving the triple pattern matching queries in big data. The Resource Description Framework (RDF) data were analyzed with Hadoop map-reduce at two levels. The table sufficient for solving the triple pattern was accessed rather than examining the entire database. The execution time of the MuSe model was lower; however, it lacked appropriate optimization, which reduced the accuracy. P. Natesan et al. [44] presented a multiple linear regression with the MapReduce programming system (MR-MLR) that exhibited constant performance for an increased subset of features. The information loss in processing big data was handled with intensive applications and predicted the target labels. The performance evaluation of MR-MLR showed improvements in the mean absolute and R-squared error; however, it had a higher complexity. The Association Rule Mining-based MapReduce framework was presented by N. Bhattacharya et al. [45] to reveal the relationship between data. The data mining techniques utilized in the model removed inconsistencies, such as missing values, redundant information, and outliers, ensuring the quality and reliability of the analyzed data. The Apriori algorithm was implemented on the Hadoop map-reduce framework, which maintained constant performance with large volumes of data. However, the lack of cloud computing systems increased the execution in real-time conditions.

K. Maheswari and M. Ramakrishnan [46] developed a Kernelized Spectral Clustering Conditional Maximum Entropy MapReduce (KSC-CMEMR) approach, which improved the clustering of large data. The KSC-CMEMR model improved the clustering accuracy while minimizing the dimensionality. Depending on the similarity index, the spectral clustering process eliminates the irrelevant information from the data; nevertheless, the model evaluated fewer data points. T. H. Sardar and Z. Ansar [47] introduced a novel MapReduce-based fuzzy C-means algorithm for big data clustering, executed over various Hadoop architecture sizes. The algorithm was developed with the Hadoop platform, which improved the clustering process's performance gain. Even though the clusters exhibited low overhead, the model failed to evaluate the quality of the clusters. R. Krishnaswamy et al. [48] presented a MapReduce hybrid density-based clustering and classification (MR-HDBCC) algorithm for big data analytics. The model harnessed density-based clustering, detecting various clusters and random shapes with the unstructured data. The cockroach swarm optimization (CSO) algorithm determined the ideal parameters for density-based clustering and showed effective performance. However, the model failed to classify large sets of data.

M.N. Akhtar et al. introduced a tipping point scheduling algorithm-based map-reducing framework [49] to optimize the workflow when executing multiple nodes. The parallel image segmentation evaluated the scheduling algorithm, which determined the execution performance. The execution time of the model was lower; however, there was a higher gap in sequential processing. A. P. Rodrigues and Niranjana N. Chiplunkar [50] developed a Hybrid Lexicon-Naive Bayesian Classifier (HL-NBC) for evaluating big data in online platforms. The classification approach utilized in the model categorized the information into diverse categories. The model showed improvements in accuracy but faced several problems in the cross-lingual classification of big data. S. Madan and P. Goswami [51] mod-

eled a k-anonymization and swarm-optimized map-reduce framework, which utilized swarm-based algorithms for maintaining privacy in cloud databases. The model derived the fitness function for maintaining higher privacy along with managing low information loss in the network. Regarding fitness, the model also developed a k-anonymization method, which improved the overall privacy; however, the effectiveness of the model was low. The Multivariate Metaphor-based Metaheuristic Glowworm Swarm MapReduce optimization (MM-MGSMO) technique was presented by S. Nithyanantham and G. Singaravel [52], who handled a wide range of information in multiple data centers. An objective function was defined for each virtual machine, and the map-reduce function identified the ideal virtual machine and performed allocation. Though the computational cost of the model was lower, the performance of maintaining data privacy was not effective.

J. Liu et al. [53] designed a novel configuration parameter tuning method based on a feature selection algorithm that optimized all the parameters of the map-reduce model. The tuning method was employed with a feature selection process and selective objective function. The utilized anisotropic Gaussian Kernel effectively determined each parameter in map-reduce, and the kernel width determined the importance of the configuration parameters. However, the running time in the Hadoop model was higher than other existing techniques.

MH Shirvani [54] uses a multi-objective algorithm for the virtual machine placement problem. Each VM runs a chunk of data placed in a physical server. Therefore, the requested applications, such as big data and MapReduce projects, need several dependent VMs to burden huge amounts of bandwidth and power consumption on network elements.

3.3.2. Spark MapReduce Model

Nada M. Mirza et al. [55] introduced the Dynamic Smart Flow Scheduler (DSFS) system for Apache Spark, providing noteworthy improvements in task optimization and enhancing resource efficiency. In addition, the DSFS system obtained significant reductions in off-heap consumption and the total number of tasks. In order to address the potential load imbalance in conventional Spark setups, causing the overloading of some nodes, the DSFS system dynamically redistributes tasks concerning the real-time load data, averting overloading and guaranteeing optimal performance. Specifically, the conventional Spark setups frequently necessitate manual intervention for scaling, resulting in inefficiencies. Finally, the DSFS system overcomes the above drawback by automatically scaling the resources considering the real-time system metrics gathered with the application of Psutil. Further, the Paramiko 3.6 (a Python library) allows for secure remote execution of commands to regulate the scaling operations. Finally, the scheduler augments the system's performance regarding scalability and sustainability.

M.S. Sanaj et al. [56] developed an effective task-scheduling approach utilizing the GA-WOA algorithm. Specifically, the task features were captured from the client's task and the features are minimized with the MRQFLDA algorithm. Further, the approach separated the large tasks into sub-tasks employing the map-reduce framework. More specifically, the tasks were proficiently scheduled via the application of the GA-WOA algorithm, with more attention provided only for the task execution and the bandwidth associated with the features for resource allocation. However, the approach failed to consider the important performance parameters including the time for sending the tasks.

H. Kadkhodaei et al. [57] modeled the Distributed Boosting-inspired ensemble classifier, which sped up the performance of the heterogeneous map-reduce approach. The diversity of the base classifiers was improved by classification, which eventually increased the entropy value. The base classifiers were then aggregated, which generated an effective output. The scalability and classification performance were effective; however, the model

exhibited lower accuracy for certain datasets. M. Farhang and F. Safi-Esfahani [4] introduced a Speculative Execution model with an ANN Algorithm (SEWANN), which was a dynamic network employed to find the straggler tasks in heterogeneous conditions. The neural algorithm utilized in the model accurately estimated the processing time of the tasks, and the effectiveness of the big data was improved by minimizing errors in processing time. The weight estimation of the model was improved; however, it lacked algorithms for computing the weights for improved performance.

J. U. Lawrancea and J. V. N. Jesudhasan [58] introduced a Parallel Clustering Anonymization Algorithm (PCAA) model that maintained the privacy of data without identity disclosure. The PCAA was scalable and achieved a better trade-off between the usability and privacy of big data. The map-reducing model parallelized the anonymization process for effectively managing the large scale of information. The intrusion in the data was identified accurately; however, it faced high compatibility issues. V. Ravuri and S. Vasundra [59] introduced a Moth-Flame Optimization-Bat optimized map-reducing model (MFOB-MP), which clustered the big data with Spark architecture. The method performed feature selection and clustering, which learns big data from diverse distributed systems. The Moth-Flame Optimization-based Bat (MFO-Bat) algorithm selected the optimal characteristics and generated a feature vector to perform ideal clustering. The model handled big data with a larger sample size, they noted that the vectors and matrices can be large because high dimensions have to be addressed, and dimensionality reduction or feature selection methods should therefore be used for better scalability and performance concerning larger datasets. X. Li et al. [60] modeled a recursively updated MapReduce-based principal component analysis (RMPCA) model, which effectively detected the errors occurring in the time-varying processes. A variable-width bin histogram increased mutual information (MI) calculation. The distributed PCA was upgraded recursively with a forgetting factor approach, and a hierarchical fault diagnosis was monitored for further processing. The computational cost of the model was decreased, yet the compatibility of the upgraded Bayesian fusion method was reduced.

N. Wang et al. [61] presented a distributed image segmentation strategy that efficiently handled composite images of larger sizes. The model generated a massive image and decomposed it into sub-images for distribution across various systems. The segmentation strategy solved the data fitting issues while processing big data; however, for complex data, the model showed lower effectiveness in processing. X. Tan et al. [62] modeled an adaptive Spark-based remote sensing data processing approach, which showed higher efficiency with map-reduce-based remote processing. The developed model achieved improved stability and efficiency in a cloud environment. A mapping and reducing strategy was employed for image tiles, and the adjacent tiles showed improvement in large volumes of remote sensing data. Nonetheless, the Spark model was limited to pixel-based classification. S. Salloum et al. [63] introduced a random sample partition (RSP) framework that represented the big data as a set of non-overlapping data subsets. Each RSP data block was employed with a probability distribution, and an efficient block sampling selected the sample data from the distributed database on a computing cluster. The RSP model was applied for various data analytical tasks, including ensemble learning and statistical estimation, yet the random subspace models required a partitioning algorithm.

K. B. Krishna et al. [64] developed a Hierarchically Distributed Data Matrix (HDM) Data Flow Optimization framework, which effectively controlled the execution of Hierarchically Distributed Data Matrix (HDM) bundles. The job completion time of the HDM data flow model was effective; however, it was not resistant to intrusions and low fault resilience to larger sets of data.

U. Ramakrishnan and N. Nachimuthu [65] introduced a memetic algorithm for feature selection in big data evaluation with the MapReduce model in which the classification performance improved the partial outcome of the model. The reduction process combined the weights of the features that defined the collection of characteristics. The classification process was improved with the support vector machine (SVM); however, the method lacked parameter tuning methods.

S. Seifhosseini et al. [66] presented the task scheduling framework for the execution of Bag-of-Tasks (BoT) in IoT applications and considered the multi-objective optimization problem as the NP-Hard problem. In addition, the model utilized the multi-objective cost-aware Discrete Gray Wolf algorithm (MoDGWA) for handling the BoT scheduling problem for the execution of IoT applications with different cost perspectives and multi-constraints. Furthermore, the approach defined the new scheduling failure factor (SFF) metric in order to evaluate resource reliability.

S. Vats and B. B. Sagar [14] modeled an independent time-optimized hybrid network that maintained data independence with the resource-sharing environment. The vertical and horizontal scaling improved the performance, and the data nodes handled large volumes of data. The time consumption of the scaling performance was reduced. However, the model was only effective with fewer nodes. L. Lamrini et al. [67] introduced a topic approach for multi-criteria decision-making that dealt with the uncertainties in the large data iteration process. The response time of the map-reduce model is improved and facilitates the robustness of a high-dimensional problem. However, the computational time of the topic approach is higher.

S. Dhamodharavadhani and R. Rathipriya [68] modeled a MapReduce-based exponential smoothing model that executed highly parallelizable algorithms over a larger database. The model utilized Holt's exponential smoothing, which showed improvements in run time compared to the continuous implementation. However, for the initialization process, the model required better optimization techniques. S. Narayana et al. [69] proposed the ACSO-enabled Deep RNN: Ant Cat Swarm Optimization-enabled (DCM, 2016) modeled Deep Recurrent Neural Network, which processed the data in a parallel manner received from distributed sources. With added Ant Lion Optimization and Cat Swarm Optimization, the input classification model became effective with added efficiency to classify data into two classes. Inspired by cat behavior, the Cat Swarm Optimization (CSO) has two modes: seeking and tracing. As cats explore looking sandbox in seeking mode they explore the search space with small movements to discover promising areas, thus exploring more. The tracing mode is more about exploitation: cats pursue the best-known cat position, refining their positions according to their own velocities/directions. The balance between exploration and exploitation efficiently helps the optimization task. For feature selection, Black Hole Entropy Fuzzy Clustering based on Pearson correlation was employed, but more data reduced the extraction quality [70].

3.3.3. Hive MapReduce Model

P. Jain et al. [8] modeled a Secured MapReduce (SMR) model that introduced a security layer between the HDFS and MapReduce, which benefits data sharing and results in the better mining of resources. The scalability issues and the trade-off between the data miners were resolved with the SMR model. The information loss that occurred during processing was reduced with the optimization during the transaction; however, in the real-world conditions corresponding to the attacks, the secure management of big data was difficult. F. Kong and X. Lin [30] introduced a MapReduce-based moving trajectory framework that reduced the issues in large data mining. The Hadoop platform was aggregated with the

mining algorithm to effectively extract data trajectory characteristics. The processing time of the model was low; however, it faced higher complexity in mining a wide range of data.

P. S. Rao and S. Satyanarayana [71] implemented the Nearest Similarity-Based (NSB) clustering for managing sensitive vulnerabilities and ensuring privacy. The bottom-up generalization achieved anonymity and computed the sensitive levels with the index values. The NSB-based model performed well with a large amount of data and achieved higher efficiency. However, the NSB model faced vulnerabilities in handling big data mining in real environments. H. Lin et al. [6] presented a map-reduce-based solution for classifying the data based on the origin. The model simplified the sequence of data into various data types and logical patterns. The implementation of MapReduce with the Pig framework improved the computational performance. The accuracy and classification performance of the model were higher for sequence data; nonetheless, the time taken for processing a single set of data was longer.

3.3.4. MongoDB MapReduce Model

A. A. Brahim et al. [72] introduced an automatic approach for the extraction of the physical system from the document-oriented NoSQL database. The system's robustness in handling the big data was higher and extracted various factors, including the structure and sources. The sequence of transformations revealed the relationship between the patterns; however, the consistent strategies in the model were not suitable for real-world conditions. T. S. Yang et al. [12] introduced a Big Data Analytics Multi-Nodal Implementation of the Apriori algorithm with a reducing model (BDAM4N), which processed both structured and unstructured databases. The model addressed the issues regarding the processing of big data and the implementation of MongoDB for the input big data storage mechanism. The BDAM4N model analyzed different data formats with high velocity and volume; however, it suffered from a delay in data processing.

3.3.5. Cassandra MapReduce Model

M. Asif et al. [11] introduced a MapReduce-Based Intelligent Model for Intrusion Detection (MR-IMID) to automate intrusion detection with multi-data classification tasks. The MR-IMID model processed the big datasets and minimized the inconsistencies in processing the big data. The ML techniques and the map-reduce model exhibited improved detection outcomes; however, the model's performance was reduced with larger datasets.

3.3.6. Pig MapReduce Model

M. Banane and A. Belangour [20] presented a Model-Driven Approach that effectively transformed the queries into big data query language scripts. The model analyzed the queries in semantic data, and the run time was lower. However, in terms of analyzing the flow of semantic data, the model experienced severe issues in efficiency.

3.3.7. Others

M. Akhtar et al. [73] introduced an IoT-based Student Health Monitoring method in which the convolutional neural network and the autoencoder effectively extracted the deep characteristics of the health information. The ideal selection of attributes is exhibited with the optimization for better extraction of parameters, and then the effectively classified health status with the weight-optimized recurrent neural network. The model yielded higher accuracy; nonetheless, the precision in classification was lower. A duplicate-divergence-different property enabled dragon Genetic (DDDG) anonymization model was presented by S. Madan and P. Goswami [74], which allowed privacy in big data by altering the map-reduce approaches. The anonymity model improved the utility performance with

the DDDG algorithm and produced better results. However, the model required several modifications regarding the optimization algorithms.

Y. Zhu et al. [75] developed a Fast Recovery MapReduce (FAR-MR) framework that improved the performance of the model in retrieving the failures. The FAR-MR model employed a fault tolerance strategy with a proactive push mechanism supporting faster recovery from node and task failures. The later progression of the failed task was obtained from the distributed storage during the recovery of failures. The distributed storage technology allowed the model to resume the computation from the previously recorded progress; however, it is vulnerable to higher task failures. A Mutual Informative MapReduce and Minimum Quadrangle Classification (MIMR-MQC) framework was developed by M. Ramachandran et al. [15] that effectively handled the limitations regarding the classification of big data. The computational complexity of the model was reduced by utilizing a big dataset, and the minimum Quadrangle support vector machine improved the efficiency of the classification process. The efficiency of the method was better when handling large-scale data; however, the evaluation was limited to a reduced range of test cases.

C. Banchhor and N. Srinivasu [76] presented a fuzzy correlative naive Bayes (FCNB) classifier that reduced the issues in big data classification. The database values are transformed into the probabilistic index table, which includes the information and attributes. The unique symbols of data in each attribute were obtained, and the classifier attained the training information for the data classes. The FCNB classifier exhibited improved classification with smaller data; however, the performance was lower for large data conditions. V. Vennila and A. R. Kannan [77] developed the linguistic fuzzy rule with the canopy MapReduce (LFR-CM) system to classify the data efficiently. The canopy MapReduce function classified the information for sharing the details in the cloud with improved accuracy. The map function was utilized concurrently without the transmission of data to other nodes. The canopy shuffle MapReduce algorithm improved the run time for the categorization of big data, and the input–output cost of the model was computed to determine the effectiveness of cloud user density. The linguistic fuzzy reduced the classification time but required generalized fuzzy Euler graphs for better classification.

3.4. Key Features of MapReduce

MapReduce offers many sophisticated features, making it more suitable for processing big data on different application areas. The key features and advantages of MapReduce are summarized in Figure 4.

An accurate example of high scalability would be Apache Hadoop MapReduce, which scales well so that it can process and store significant amounts of data across many cheap, not-so-powerful servers independently. We simply need to add more servers or nodes to increase storage and computational capabilities. This architecture can scale to process applications on thousands of nodes handling petabytes of data. MapReduce enables the processing of different types of data sources and thus can be utilized for various kinds of business requirements. Both structured and unstructured data are supported, allowing enterprises to capture insights on multiple data streams. Hadoop works on the principle of an open-source platform that allows enterprises to modify the code to make it more compatible with their specific requirements. The security model is strong, taking advantage of HBase and ease-of-use protocols for data in MapReduce. It replicates every data block on multiple machines to achieve fault tolerance, thus making it possible to recover data from other nodes when some of the hardware has failed. MapReduce allows the processing of large datasets and has an architecture to store the big data, which means it provides a cost-effective solution when we need lacs or millions of storage capabilities. It also makes it ideal for businesses that are managing ever-expanding data, all with relatively

low costs. The system achieves its speed through the Hadoop Distributed File System (HDFS), allowing MapReduce to process large amounts of data in parallel on a distributed network of CPUs. This system, in turn, facilitates large-scale data processing at a high speed, making MapReduce ideal for quickly examining massive datasets. MapReduce is based on a simple programming model that simplifies the development of applications capable of processing vast amounts of data. The popular Java programming language creates MapReduce applications, making it accessible to many developers. It inherently supports parallel processing, executing tasks concurrently across multiple processors. This capability enhances data processing efficiency, significantly reducing the time required to handle large datasets. Data reliability is assured in MapReduce through redundancy; data blocks are replicated across multiple nodes to prevent data loss in case of node failure. Several data verification modules within the Hadoop ecosystem provide additional reliability. MapReduce's architecture is designed for high availability with redundant components such as the Name Nodes in the Hadoop cluster. This setup ensures continuous data access even if one of the critical components fails.

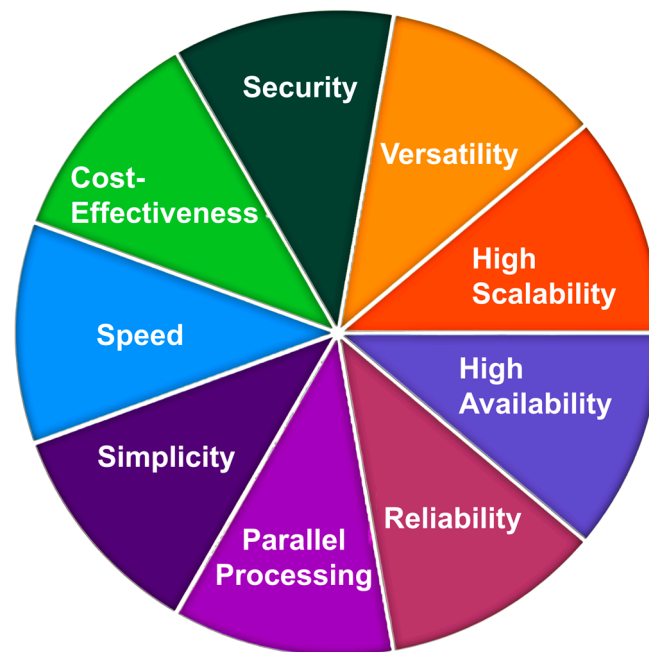


Figure 4. Key features of MapReduce.

3.5. Applications of MapReduce

MapReduce is extensively used in various sectors; it is one of the most essential tools of the trade, and it is used in various sectors for the processing and analyzing of large datasets. This enables it to process massive amounts of data in parallel on many computers more quickly and at greater scales. This illustration demonstrates some industries using MapReduce to improve how they work and innovate in their industries. Whether it is entertainment platforms like Netflix studying viewer habits or financial institutions identifying fraudulent activity, MapReduce is a fundamental force in digital ecosystems. It finds use cases in the domain of e-commerce, social media, and data warehousing and has become a must-have tool for every company looking to recommend products better, understand social media movement, or store large amounts of data. This visualization highlights how critical and versatile MapReduce is to powering Business Analytics and Operational Intelligence from a broad range of domains, the major ones of which can be seen in Figure 5.

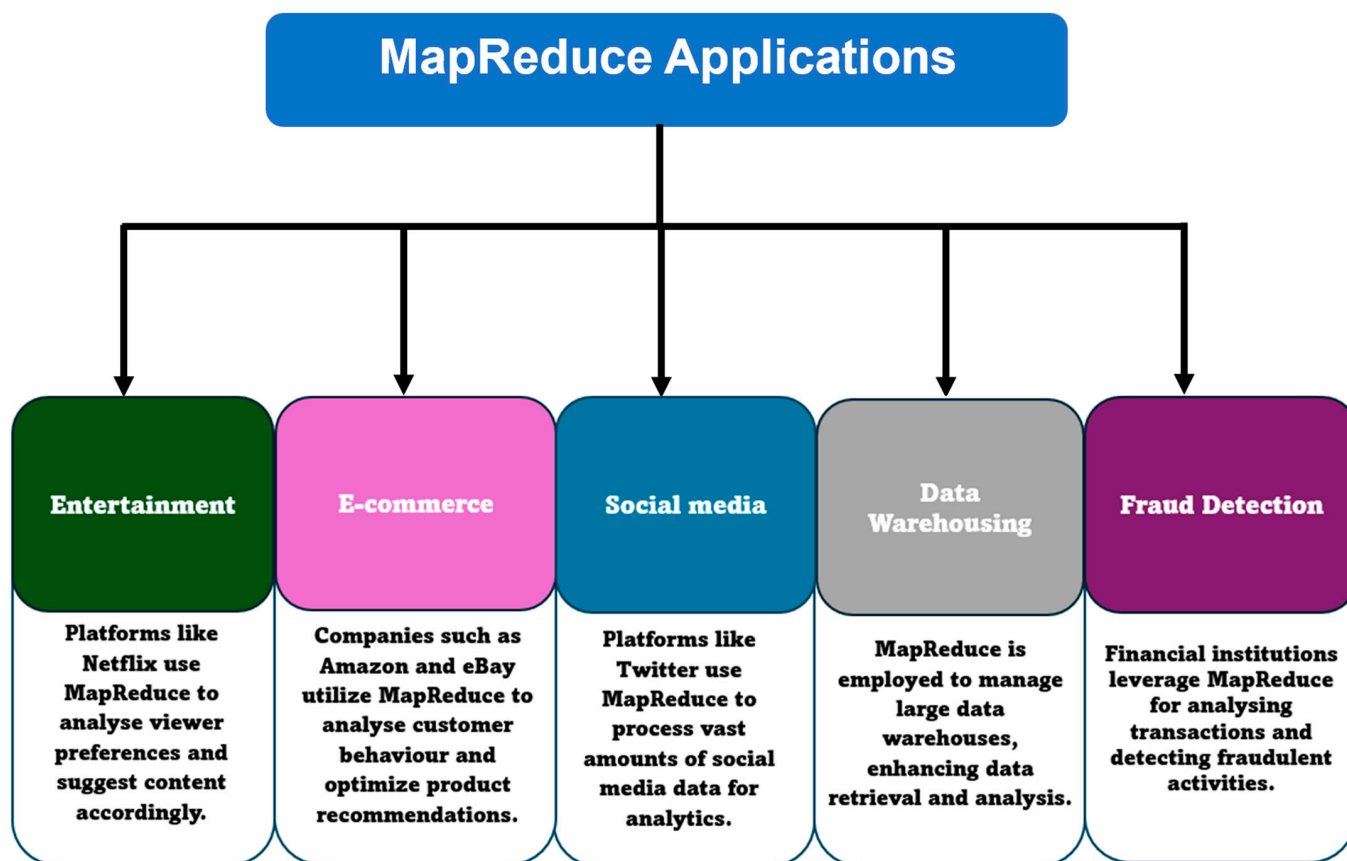


Figure 5. Applications of MapReduce.

3.6. Overall Analysis of the MapReduce Models

The overall evaluation of the reviewed literature with the diverse utilized methods, their advantages, achievements, and their limitations are discussed in Table 1.

Table 1. Comparison of the MapReduce models.

S. No	Authors	Year	Method	Dataset	Achievements	Pros	Cons
1	Issa Mohammed Saeed Ali and Dr. D. Hariprasad [2]	2023	MRHPOELM	Higgs dataset, glass, Diabetes, Cleveland, Vehicle, and Wine dataset	Higgs dataset: Accuracy 99.32%, precision 96.12, recall 97.64%, F-measure 96.87%, specificity 97.11%, and time 51.09 s	The hybrid feature selection minimized the imbalance issues	Failed to manage multi-source data
2	Ankit Kumar et al. [1]	2023	Hadoop map-reducing framework	Hadoop	Data processing time by 30%	The output key and the value pairs were reset for the effective retrieval of a wide range of information	Incompatibility of diverse nodes
3	DEVENDRA DAHIPHALE [21]	2023	Distributed and parallel algorithm for 2-ECCs BiECCA	-	Processing time 144 s for 8 input records	The vast database enabled the stream processing of data	The small and large star process manages only one edge at a time
4	Md. Mobin Akhtar et al. [73]	2023	IoT-based Student Health Monitoring System	Healthcare dataset	Accuracy 94.02%, precision 89.73%, False positive rate (FPR) 6.44, F1 score 92%	The auto-encoder and one-dimension convolutional neural network effectively extracted the deep characteristics	Low precision
5	M.S. Sanaj and P.M. Joe Prathap [56]	2021	M-CWOA	Healthcare dataset	-	Features are effectively reduced with the application of MRQFLDA algorithm.	The model ignored the important performance parameters
6	Neha Verma et al. [22]	2020	GA-WOA	-	Processing time 55.68 ms, map process execution time 31.5 ms, virtual memory 25.08 bytes	The Apriori algorithm employed with cloud architecture led to the effective mining of data.	Lower data handling ability
7	So-Young Choi and Kyungyong Chung [3]	2020	Hadoop platform-based MapReduce system	Peer-to-peer dataset	Recall 67.39%, precision 81.12%, F-measure 73.62%	The reduction in key values improved the processing time	Limited to processing unstructured data
8	Tanvir H. Sardar and Zahid Ansar [23]	2022	MapReduce-based Fuzzy C-Medoids	20_newsgroups	Execution time 8 node: 285 s, performance ratio 1.31	The utilized algorithm was scalable with large-scale datasets	The quality of the clusters was not evaluated
9	Priyank Jain et al. [8]	2019	SMR model	Twitter dataset	Information loss 5.1%, running time 12.19 ms	The difference in processing time minimized with a larger data size	The security of the big data was not maintained in real-world conditions.
10	Surendran Rajendran et al. [19]	2021	CPIO-FS	ECBDL14-ROS dataset, epsilon dataset	Training time 318.54 s, AUC 72.70%	The HHO-based deep belief network (DBN) model improved the classification performance.	Limited quality of service parameters

Table 1. Cont.

S. No	Authors	Year	Method	Dataset	Achievements	Pros	Cons
11	Shaopeng Guan et al. [24]	2024	The Hadoop-based secure storage scheme	Cloud big data	encryption storage efficiency-improved by 27.6%	The dual-thread encryption mode improved the speed of encryption	Required advanced computing framework for the efficient calculation of cipher text.
12	Sumalatha Saleti, Subramanyam R.B.V. [10]	2019	CRMAP	Kosarak, BMSWebView2, MSNBC, C20 – D2 – N1K – T20	Minimum Support 0.25%, execution time 25 s	Resolved the sequential pattern issues of big data	The map-reducing framework was limited to a single data-processing model
13	Thanh Dat Dang et al. [25]	2019	Trust-based MapReduce framework	-	Execution time 5.5 s	Low data processing costs	Lower security in processing
14	Hamidreza Kadkhodaei et al. [57]	2021	Distributed Heterogeneous Boosting-inspired ensemble classifier	CEN, EPS, COV, POK, KDD, SUSY, HIG	Classification accuracy of 75%, performance of 16 for 32 mappers, and scalability	The performance of the distributed heterogeneous ensemble classifier was effective.	Lower accuracy for certain datasets
15	G. Venkatesh and K. Arunesh [9]	2019	Layers 3 traffic-aware clustering method	Reuters-21578, 20 Newsgroup, Web ace, TREC	Accuracy 89%, Execution time 14.165 s	Improved accuracy with the three-layered structure	Higher execution time
16	Junghee Jo and Kang-Woo Lee [26]	2019	DELT	Marmot	Data preparation time 116 s	Smaller performance degradation with optimization	Poor performance in dealing with spatial queries with geospatial big data
17	K. Meena and J. Sujatha [27]	2019	Co-EANFS	UCI datasets	Accuracy 96.45%, error rate 3.55%	The success ratio of the recommender system was higher	Required regression models for effective performance
18	Omkaresh Kulkarni et al. [28]	2020	FrSparse FCM	Skin dataset and localization dataset (UCI)	Accuracy 90.6012%, DB Index 5.33	The sparse algorithm effectively computed the optimal centroid in the mapper phase.	Lower classification accuracy
19	Hossein Azgomi, Mohammad Karim Sohrabi [29]	2021	MR-MVPP	TCP-H dataset, TPC BENCHMARK H	A coverage rate of 60% for 5 queries	Better performance in terms of coverage rate	The responsive time for the analytical queries in online platforms was higher.
20	Dai-Lun Chiang et al. [30]	2021	Petri net model	Corporate datasets	-	The file and rule checker reduced the errors in the analysis system.	No real-time performance evaluation

Table 1. Cont.

S. No	Authors	Year	Method	Dataset	Achievements	Pros	Cons
21	Suman Madan and Puneet Goswami [74]	2019	DDDG anonymization	UC Irvine dataset	Classification accuracy 89.77%	The DDDG algorithm modifies the map-reduce technique and preserves data privacy.	Needed modifications in the optimization algorithms
22	Yongqing Zhu et al. [75]	2020	FAR-MR	Apache Hadoop YARN	Performance time 500 s	Improved performance in task failure recovery	Higher task failures
23	Mandana Farhang and Faramarz Safi-Esfahani [4]	2020	SWANN	Apache, W.E	Execution time	The neural network algorithm accurately estimated the execution time of the tasks.	Limited in learning algorithms for computing weights
24	Felwa Abukhodair et al. [31]	2021	MOBDC-MR	Healthcare dataset	Computational time 5.92 s, accuracy 92.86%	The hyperparameter tuning of the LSTM model enhanced the performance of the classifier.	The classification accuracy of the system was lower.
25	Manikandan Ramachandran et al. [15]	2021	MIMR-MQC	CBTRUS dataset	Detection time 40 ms, accuracy 90%, computational complexity 40 ms	Better handling of large-scale data	Reduced range of test cases
26	Chitrakant Banchhor and N. Srinivasu [76]	2019	FCNB	Skin segmentation and localization dataset	Sensitivity 94.79%, specificity 88.89%, accuracy 91.78%	Low time complexity	Classification performance was lower for large data conditions
27	Josephine Usha Lawrancea and Jesu Vedha Nayahi Jesudhasan [58]	2021	PCAA	Shlearn dataset	Execution time 40 ms, completion time 88 ms	Map-reduce and ML technologies identified the intrusions	Higher compatibility issues
28	V. Vennila and A. Rajiv Kannan [77]	2019	LFR-CM	Imbalance DNA dataset	Input/output cost 35 ms, classification accuracy 90%, classification time 20 ms, run time 30 ms	The linguistic fuzzy reduced the classification time	Generalized fuzzy Euler graphs were needed for better classification
29	Fansheng Kong and Xiaola Lin [70]	2019	Map-reduce-based moving trajectory	MapReduce and HBase distributed database	-	Trajectory characteristics were obtained with the mining algorithm	Higher complexity in data mining
30	Ankit Kumar et al. [78]	2020	Replication-Based Query Management system	High indexing database	Execution time 40 ms, completion time 88 ms	The execution time of the Hadoop architecture was lower	Higher complexity issues during processing
31	Dr. Prasadu Peddi [5]	2019	HDFS	Stocks dataset	-	Better performance with unstructured log data	Higher replications of block data
32	Deepak Kumar and Vijay Kumar Jha [17]	2021	NKM	Hospital Compare Datasets	Execution time 350 ms, retrieval time 20,000 mb, F-measure 85%, precision 96%, recall 92.5%	The query optimization process enhanced the performance with larger datasets.	Higher retrieval time

Table 1. Cont.

S. No	Authors	Year	Method	Dataset	Achievements	Pros	Cons
33	Zheng Xu [32]	2022	CICC-BDVMR	Skin segmentation dataset	Accuracy 82.80%, TPR 86.27%, TNR 79.06%	Map-reduce and ML technologies identified the intrusions	The detection outcome was lesser with more data
34	Mazin Alkathiri [33]	2019	Multi-dimensional geospatial data mining based on Apache Hadoop ecosystem	Spatial dataset	Processing time 140 s	Big geospatial data processing and mining	Lacks in visualization interface
35	Abolfazl Gandomi et al. [34]	2019	HybSMRP	-	Completion time 103 s	Delay scheduling reduced the number of local tasks	The effectiveness of the scheduler was not evaluated with larger datasets
36	Muhammad Asif et al. [11]	2022	MR-IMID	NSL-KDD dataset	Training and validation detection accuracy 97.6%, detection miss rate 2.4%	Map-reduce and ML technologies identified the intrusions	The detection outcome was lesser with more data
37	Amal Ait Brahim et al. [72]	2019	Automatic model-driven extraction	K-anonymization based dataset	Information loss 1.33%, classification accuracy 79.76%	Higher data privacy with optimization	Low effectiveness
38	Chitrakant Banchhor and N. Srinivasu [35]	2022	HCNB-MRM	Localization dataset and skin dataset	Accuracy of 93.5965%, and 94.3369%	The map-reduce framework parallelly handled the data from distributed sources.	Not evaluated with deep learning approaches
39	P. Srinivasa Rao and S. Satyanarayana [71]	2018	NSB	A real-world GPS trajectory dataset	-	The Apriori algorithm effectively mined large amounts of data	The absence of cloud computing systems increased the execution in real-time conditions
40	Suman Madan and Puneet Goswami [51]	2020	K-anonymization and swarm-optimized map-reduce framework	K-anonymization based dataset	Information loss 1.33%, classification accuracy 79.76%	Higher data privacy with optimization	Low effectiveness
41	J. Ramsingh and V. Bhuvaneswari [37]	2021	NBC-TFIDF	-	Execution time 54 s, precision 72%, recall 75%, F-measure 73	Higher performance with the multi-mode cluster	Low precision
42	Khaled Alwasel et al. [16]	2021	BigDataSDNSim	-	Completion time 158.96 s, transmission time 26.98 s, processing time 130 s	Seamless task scheduling	High transmission delay in cloud environments
43	Dawen Xia et al. [38]	2021	WND-LSTM	A real-world GPS trajectory dataset	MAPE 65.1%, RMSE 1.4, MAE 0.10, ME 0.44	The efficiency of short-term traffic prediction was improved	The accuracy and scalability of the model were not evaluated

Table 1. Cont.

S. No	Authors	Year	Method	Dataset	Achievements	Pros	Cons
44	S. Nithyanantham and G. Singaravel [52]	2021	MM-MGSMO	Amazon EC2 big dataset	Computational cost, storage capacity 45 mb, FPR 12.5%, data allocation efficiency 90%	Processed massive data by selecting cost-optimized virtual systems	Low data privacy
45	Vasavi Ravuri and S. Vasundra [59]	2020	MFOB-MP	Global Terrorism Database	Classification accuracy of 95.806%, Dice coefficient of 99.181%, and Jaccard coefficient of 98.376%	Handled big data with large sample size	High dimensionality characteristics were not considered
46	Han Lin et al. [6]	2018	Map-reduce classification	-	-	Parallel clustering approach improved the potential of big data clustering	Load balancing issues
47	Terungwa Simon Yange et al. [12]	2020	BDAM4N	NHIS data	Response time 0.93 s, throughput 84.73%	Processed both structured and unstructured database	Delay in processing data
48	Xintong Li et al. [60]	2020	RMPCA	-	Computational cost 0.250 s	Detected the errors in time-varying processes	Reduced compatibility of the updated Bayesian fusion method
49	Ning Wang et al. [61]	2020	A distributed image segmentation strategy	-	Response time 0.93 s, throughput 84.73%	Efficiently refined the raw data before training	Low effectiveness in the context of complex data
50	Xicheng Tan et al. [62]	2021	Adaptive Spark-based remote sensing data processing method	Resilient distributed dataset (RDD)	Execution time 45 Vms	Higher efficiency with map-reduce-based remote processing	Limited to pixel-based classification
51	Salman Salloum et al. [63]	2019	RSP	RDD	Execution 12 s	The probability distribution in data subsets was similar for the entire dataset	Necessitated partitioning algorithm for random subspace models
52	K BALA KRISHNA et al. [64]	2019	HDM	RDD	Time consumption 10 ms	Better execution control of HDM bundles	Low fault resilience
53	Celestine Iwendi et al. [7]	2019	Temporal Louvain approach with TF/IDF Algorithm	Reuters-21578 and 20 Newsgroups	Execution time 69.68 s, accuracy 94%	The complicated structure of the streaming data was effectively obtained	Higher execution time for complex data
54	JUN LIU et al. [53]	2020	Novel configuration parameter tuning based on a feature selection algorithm	-	Execution time 32.5 s	The K-means algorithm amplified the performance of parameter selection	Higher running time in the Hadoop model
55	Umanesan Ramakrishnan and Nandhagopal Nachimuthu [65]	2022	Memetic algorithm for feature selection in big data analytics with MapReduce	Epsilon and ECBDL14-ROS datasets	AUC 0.75	The classification process improved the partial outcome	Lacked parameter tuning

Table 1. Cont.

S. No	Authors	Year	Method	Dataset	Achievements	Pros	Cons
56	Namrata Bhattacharya et al. [45]	2019	Association Rule Mining-based MapReduce framework	Comma-separated file	Transaction time 140 ms	The Apriori algorithm effectively mined large amounts of data	The absence of cloud computing systems increased the execution in real-time conditions
57	K. Maheswari and M. Ramakrishnan [46]	2021	KSC-CMEMR	HTRU2 dataset and Hippar cos star dataset	Clustering accuracy 77%, clustering time 45 ms, space complexity 44 MB	KSC-CMEMR improved the clustering of large data with minimum time	Fewer data points were evaluated
58	Tanvir H. Sardar and Zahid Ansar [47]	2022	Novel MapReduce-based fuzzy C-means for big document clustering	20_newsgroups dataset	Execution time 108.27 s	Low cluster overhead	The quality of clusters was not evaluated
59	R. Krishnaswamy et al. [48]	2023	MR-HDBCC	Iris, Wine, Glass, Yeast-1 dataset, Yeast-2 dataset	Computational time 11.85 s, accuracy 95.43%, formation of clusters 25.42 ms	Effective performance on density-based clustering	Large sets of data were not classified
60	Seifhossein, et al. [66]	2024	MoDGWA	BoT-IoT dataset	The model attained a reduction of 0.55%, 7.28%, 10.20%, and 45.83% over other existing models in terms of makespan, ToC, TSFF, and the cost score.	Reduced the execution cost and improved the overall reliability	Require improving the efficiency
61	Nada M. Mirza [55]	2024	Dynamic Smart Flow Scheduler	Real-time load data	The average resource consumption was minimized by 19.9%, from an average of 94.1 MiB to 75.6 MiB, accompanied by a 24.3% reduction in the number of tasks	The model offers more effective and responsive distributed data-processing environment	Required enhancing the scalability
62	Mohammad Nishat Akhtar et al. [49]	2020	Tipping point scheduling algorithm-based map-reducing framework	Image dataset	Running time 60 s for 2 nodes, data processed 12/s	Higher substantial speed for more number of nodes	A higher gap in sequential processing
63	S. Dhamodharavadhani and R. Rathipriya [68]	2019	MapReduce-based Exponential Smoothing	Indian rainfall dataset, Tamil Nadu state rainfall dataset	MSE accuracy 83.8%	Highly parallelizable algorithms were executed over large databases	Optimization was required for a better initialization process
64	Satyala Narayana et al. [69]	2022	ACSO-enabled Deep RNN	Cleveland, Switzerland, and Hungarian datasets	Specificity 88.4%, accuracy 89.3%, sensitivity 90%, threat score 0.827	Data from the diverse distributed sources were managed in a concurrent way	Lower privacy in healthcare datasets
65	Anisha P. Rodrigues and Niranjan N. Chiplunkar [50]	2022	HL-NBC	Hadoop, Flume, Hive datasets	Accuracy 82%, precision 71%, recall 70%, F-measure 71%	Topic classification classified the texts effectively	Cross-lingual sentimental classification issues

Table 1. Cont.

S. No	Authors	Year	Method	Dataset	Achievements	Pros	Cons
66	Mouad Banane and Abdessamad Belangour [20]	2020	Model-Driven Approach	LUBM1, LUBM3, LUBM2 datasets	Runtime 400 ms	The driven engineering-based approach effectively transformed the queries into Hive, Pig, or Spark scripts.	Efficiency issues in processing semantic data flow
67	Wuhui Chen et al. [13]	2020	Map-reduce-based QoS data placement technique	Microsoft Bing's data centers	Execution time 10,000 s, communication cost 5000 MB	Minimized the joint data placement issues in map-reduce applications	Communicational costs and higher energy costs
68	Ashish Kumar Tripathi et al. [39]	2020	MR-MDBO	Susy, pkerhand, DLeOM, sIoT, IoT-Bonet	Computation time 9.10×10^3 , F-measure 0.846	The speed analysis of the algorithm exhibited higher scalability	Not evaluated with real-world applications of big data
69	Neda Maleki et al. [40]	2020	TMaR	Amazon EC2 big dataset	Completion time 1500 s, time duration 110 s, relative start time 150 s	Reduced the makespan with network traffic	No parallelism between map and reduced tasks
70	Satvik Vats and B. B. Sagar [14]	2020	Independent time-optimized hybrid infrastructure	Newsgroup dataset	Time consumption 41 s	Data independence with sharing of resources	Effective with fewer data nodes
71	Loubna Lamrini et al. [67]	2023	Topsis approach for multi-criteria decision-making	Generated dataset, mobile price dataset, credit card clients dataset	Execution time 90 s	Dealt with uncertainties in iterations and larger datasets	Longer computation time
72	Uma Narayanan et al. [41]	2022	SADS-Cloud	-	Information loss 0.02%, compression ratio 0.06%, throughput 7 mbps, encryption 0.07 s, decryption time 0.03 s, efficiency 58.12 s	Managed big data over a cloud environment	Encryption and decryption issues
73	R. Thanga Selvi and I. Muthulakshmi [42]	2021	MRODC	Pima Indian dataset	Precision 91.8%, recall 90.89%, accuracy 88.67%, F1 score 91.34%	Improved clustering efficiency	No real-time diagnosis of patient information
74	Tanvi Chawla et al. [43]	2021	MuSe	LUBM and WatDiv	Execution time 2500 ms	Effectively solved triple pattern matching queries in big data	Low SPARQL optimization process
75	P. Natesan et al. [44]	2023	MR-MLR	MSD dataset	Mean absolute 20, R-squared error (R2error) 0.90, RMSE 200	Constant performance even for an increased subset of datasets	Higher complexity

4. Analysis Based on the Utilized Datasets

The datasets utilized in the map-reducing models were analyzed and are shown in Figure 6. The analysis reveals that the Cloud dataset is the most utilized in the reviewed papers presented for map reduction. The skin and localization dataset and the newsgroup datasets were the next predominantly utilized, while the databases, including MSD, Pima Indian, Amazon EC2, Rainfall, and NHID datasets, were the least harnessed. LUBM and Healthcare datasets are also utilized in many approaches.

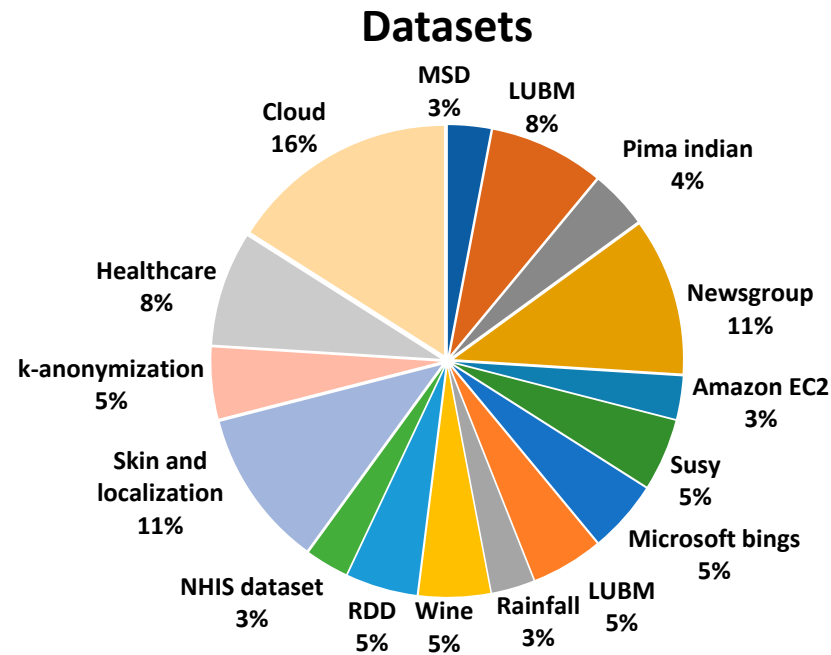


Figure 6. Evaluation based on the datasets utilized in the reviewed literature.

5. Evaluation Based on the Performance Metrics

The discussion of the performance metrics in the reviewed works, namely execution time, precision, recall, accuracy, F-measure, and speedup, are covered in this section. The metrics, including specificity, false positive rates, information loss, and several other metrics, are harnessed to evaluate the performance of the map-reduce systems, which is represented in Figure 7. The analysis reveals that execution time and accuracy are the widely used metrics in the reviewed literature, where accuracy is the most used metric to evaluate the effectiveness of the map-reduced models in managing big data.

5.1. Overview of Performance Metrics

The evaluation relies on widely accepted metrics in distributed computing research:

- Accuracy: The proportion of correctly classified instances representing the model's reliability in predicting true outcomes.
- Precision: The ratio of true positives to the total predicted positives, indicating the accuracy of positive predictions, which is especially important when false positives carry a high cost.
- Recall: The sensitivity of the model, showing the proportion of true positives among actual positives. High recall is valuable in applications where identifying all relevant instances is crucial.
- F1 Score: The harmonic mean of precision and recall, providing a balanced metric when there is an imbalance in class distributions.

- Jaccard Index: A similarity measure between predicted and actual classifications, where a higher index indicates better overlap between predicted and actual results.
- Response Time: The time taken by the system to respond to requests, a critical metric for real-time applications where low latency is preferred.
- Execution Time: The overall time taken for the task's completion, critical in large-scale applications where processing time can impact operational efficiency.
- Information Loss: Measures any degradation in data quality that may arise from distributed processing. Lower information loss suggests better retention of original data integrity.
- Specificity: Indicates the proportion of true negatives among all actual negatives, relevant in scenarios where avoiding false positives is important.

Performance metrics

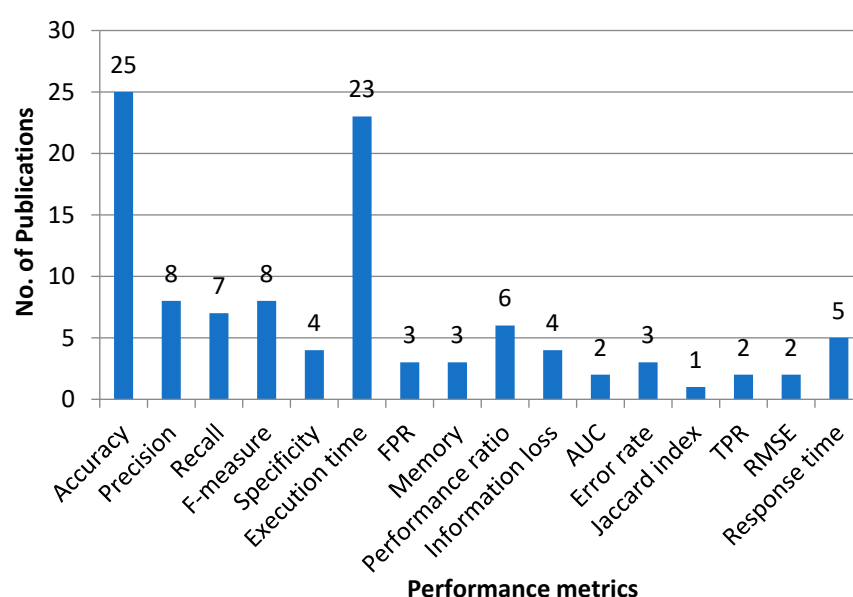


Figure 7. Interpretation based on the performance metrics utilized in the reviewed works.

These metrics collectively form the basis for a robust evaluation of the MapReduce framework's performance across various configurations.

5.2. Detailed Analysis of Metrics

5.2.1. Accuracy

Accuracy describes the standard or the quality expressing the correctness of the system, which is mathematically signified as

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

Here, FP denotes the false positive, TP is the true positive, FN is the false negative, and TN represents the true negative.

5.2.2. F-1 Score

The F-measure value or F-1 score is analyzed in the uneven data distribution and can be calculated using the values of precision and recall. The F-1 score can be mathematically represented as

$$F1 - score = 2 \times \frac{precision \times recall}{precision + recall} \quad (2)$$

5.2.3. Precision

Precision denotes the highest level of closeness obtained by the model for the same processing. This represents the ratio of accurately evaluated detections to the total predicted values which can be expressed as

$$Precision = \frac{TP}{TP + FN} \quad (3)$$

5.2.4. Jaccard Index

The Jaccard index defines the resemblance index between the two sets of information. The Jaccard similarity of the sets can be denoted by

$$Jaccard\ index = \frac{|A \cap B|}{|A \cup B|} \quad (4)$$

$|A \cap B|$ denotes the intersection of the sets A and B ; $|A \cup B|$ is the union of the sets A and B .

5.2.5. Response Time

The response time of the model is the time duration of the model to respond to the request. Response time can be signified as

$$Responsetime = \frac{N}{shareddata} - T_{think} \quad (5)$$

Here, N represents the number of processed data, and T_{think} indicates the time taken to respond to the queries.

5.2.6. Recall

Recall denotes the proportion of correctly recommended data with the total useful recommendations which can be represented by

$$Recall = \frac{TP}{TP + FN} \quad (6)$$

5.2.7. Execution Time

The execution time is the total time required for a system to complete specific tasks. It is the processing time for different types of map tasks depending on the size of the dataset.

5.2.8. Information Loss

Information loss I_L describes the losses that occur while processing large sets of data, which is denoted as

$$I_L = \frac{(original - randomized\ value)^2}{(original + randomized\ value)} \quad (7)$$

5.2.9. Specificity

The specificity of the system is identified with the number of correctly identified clusters from the negative class and can be signified as

$$Specificity = \frac{TN}{(TN + FP)} \quad (8)$$

The reviewed MapReduce models on big data evaluate several other metrics, including FPR, memory, performance ratio, AUC, error rate, TPR, and RMSE.

5.2.10. Aggregate Performance Score (APS)

APS is a single score to evaluate the aggregate performance of classification metrics across all classes together. APS considers the model's prediction power (accuracy, precision, recall, F1 score, and Jaccard Index) and the computational efficiency of the system (response time and execution time). APS penalizes for information loss and rewards for specificity to balance the performance evaluation. Such a formula allows for a more comprehensive evaluation, as in situations where both accuracy and efficiency are important.

$$APS = w_1 \times \text{Accuracy} + w_2 \times \text{F1-score} + w_3 \times \text{Precision} + w_4 \times \text{Jaccard Index} - w_5 \times \text{Response Time} + w_6 \times \text{Recall} - w_7 \times \text{Execution Time} - w_8 \times \text{Information Loss} + w_9 \times \text{Specificity} \quad (9)$$

Here, weights (w_1 to w_9) have been applied just based on the application, typically showing the importance of each metric. Positive scores (such as runtime and execution time) are deducted to incentivize the performance of your computation, while negative scores (such as information loss) decrease the score by the number of misclassifications. The APS does not penalize the model so heavily by weighting negatives because specificity rewards it for differentiating negative examples correctly, thus making it a robust and flexible metric in practice (see Table 2 for additional visual properties covered).

Table 2. Performance metrics of the classification model on the twenty+newsgroups dataset.

Metric	Value
Accuracy	0.707
Precision	0.718
Recall	0.697
F1 Score	0.694
Jaccard Index	0.542
Response Time	0.014
Execution Time	0.014
Information Loss	0.293
Specificity	0.697
APS	3.734

Table 2 shows the performance metrics of the model on the twenty+newsgroups dataset for our framework, a fine-grained evaluation of both quality and speed. A score of 0.707 implies that the samples were classified correctly at a rate of seventy percent. In the same way, precision is 0.718, meaning that out of all predicted positive samples, only 72% were actually positive while recall is 0.697, which means our model managed to capture almost 70% of relevant samples. F1 is the harmonic mean of precision and recall = 0.694. This score helps interpret how balanced our model is concerning precision and recall. The Jaccard index gives an overall idea about how much the predicted labels overlap with actual labels; specifically, it is useful in multi-label classification (like this). The value of 0.542 indicates a medium overlap between predictive and true labels.

Considering the efficiency aspect, the response time and execution time both being 0.014 s is low enough for a real-time application. The information loss value of 0.293 calculated from $1 - \text{Accuracy}$ represents the number of samples that are misclassified in the dataset. With a specificity of 0.697, this observation indicates the model's ability to accurately label negative samples. Lastly, the Aggregate Performance Score (APS) of 3.734 combines all these numbers as one score that nicely balances predictive power, computational speed, and penalties (the lower the APS value the better).

Its performance on TAR demonstrates the model is robust for large-scale text classification problems, particularly when high-throughput response times are needed and low error rates of actual classifications are not as critical.

5.3. Speedup

Speedup refers to the ratio of time consumed by the same task running in a single processor system and a parallel processor system, which is used to measure the performance and effectiveness of parallelization of parallel systems or programs. The formula for calculating the speedup is

$$\text{Speedup} = \frac{\text{Response Time (Single - Node)}}{\text{Response Time (Multi - Node)}} \quad (10)$$

Here, T_1 is the running time under a single processor, and T_p is the running time under a parallel system with p processors.

In this section, we focused on the effect of feature counts on speedup and processing time. Comparing response times (250, 500, and 750 features), we calculated speedup accurately relative to the single-node baseline. Each configuration used to compute metrics, including accuracy, precision, recall, F1 score, Jaccard index, and response time, was recalculated. The relative speedup, from single-node to multi-node response time, is shown in Table 3.

Table 3. MapReduce performance in a multi-node configuration.

Feature Count	Accuracy	Precision	Recall	F1 Score	Jaccard Index	Response Time (s)	Speedup
250	0.428	0.482	0.420	0.402	0.261	1.829	0.007
500	0.601	0.621	0.591	0.586	0.424	1.168	0.014
750	0.672	0.681	0.662	0.657	0.501	1.117	0.016

We need to proceed with the following steps:

1. Re-evaluate Speedup with Varying Feature Counts: How computational load impacts speedup.
2. Increase Sample Size for Testing: To simulate larger dataset handling and observe performance changes.

5.3.1. Key Observations

1. Speedup and Feature Count: As the feature count increases from 250 to 750, response time slightly improves. However, the speedup remains modest across feature counts, with 250 features yielding a speedup of 0.007 s, 500 features achieving 0.014 s, and 750 features reaching 0.016 s. These results indicate that increasing feature complexity in a multi-node setup does not significantly enhance speedup, as the processing overhead of handling additional features counteracts the potential gains.
2. Accuracy and Feature Complexity: Increasing the feature count generally improves classification accuracy and other metrics. For instance, with 250 features, the model attains an accuracy of 0.428, whereas, with 750 features, accuracy increases to 0.672; this illustrates the balance between feature complexity and accuracy, as higher feature counts capture more data details, improving classification quality.
3. Performance Implications: Lower feature counts result in higher speedup but reduced accuracy and precision. This trade-off suggests that while a minimal feature set may be suitable for speed-focused applications, tasks requiring high accuracy and moderate speedup would benefit from feature counts around 500–750, achieving a balance between computational efficiency and classification quality.
4. Optimal Feature Configuration for Speedup: The analysis suggests a feature count of 500–750 offers a reasonable trade-off between processing time and classification performance. While a smaller feature set can enhance response time, it adversely impacts

model accuracy. Thus, applications requiring speed and quality should consider a moderate feature count to maintain performance stability in MapReduce tasks.

5. **Consistency Across Nodes:** The classification accuracy, precision, recall, F1 score, and Jaccard index remained stable across single-node and multi-node configurations; this suggests that distributing data across nodes did not impact classification performance when the feature count was optimized. This stability underscores that the model effectively retains classification quality even with distributed data.
6. **Response Time and Speedup Limitations:** Although MapReduce is designed to expedite data processing, the speedup in this configuration is limited. Even at 750 features, the speedup remains at 0.016 s, indicating that the data distribution and coordination overhead constrain the benefits of parallel processing; this reflects the reality that multi-node setups cannot achieve the ideal linear speedup due to significant parallel computing and communication overheads, which should be a focus in future research to optimize MapReduce efficiency.
7. **Scalability Implications:** The findings highlight that increasing nodes or features alone may not significantly improve response times as dataset complexity grows. Instead, a scalable approach in MapReduce applications should consider optimized feature counts and efficient node utilization to balance processing speed and classification quality in large-scale data processing tasks.

The above results reinforce that optimizing speedup in MapReduce applications involves balancing feature complexity with response time, which is crucial for real-time big data processing.

5.3.2. Speedup with Node Scaling

Starting with a single-node baseline response time, we conducted experiments to estimate response times for multi-node configurations involving two, four, and eight nodes. These results illustrate the impact of scaling on response time and speedup, revealing the typical diminishing returns observed in distributed systems as node count increases.

Table 4 demonstrates the effect of increasing the number of nodes on response time and speedup.

Table 4. Speedup analysis with increasing node counts.

Node Count	Single-Node Time (s)	Multi-Node Time (s)	Speedup
2	0.0205	1.8087	0.0114
4	0.0728	1.4693	0.0496
8	0.0599	1.7569	0.0341

Further, each node configuration is explained as follows:

- **Single-Node Baseline:** The single-node baseline configuration is the reference point for speedup calculations. The initial response time, indicated in this table, is considered the benchmark.
- **Two-Node Configuration:** In the two-node setup, the response time is approximately 1.8087 s, and the corresponding speedup is 0.0114; this indicates some benefits from distributed processing but is not near ideal, suggesting initial overhead impacts the gains.
- **Four-Node Configuration:** Expanding to four nodes, the response time reduces to about 1.4693 s, yielding a speedup of 0.0496. Although the performance improves, the speedup achieved is still far from an ideal scaling factor, likely due to coordination costs.

- **Eight-Node Configuration:** With eight nodes, the response time is 1.7569 s, with a speedup of 0.0341. The results reflect diminishing returns, with performance gains becoming less significant as the number of nodes increases. The overhead of communication and synchronization among nodes becomes more pronounced at this scale.

5.4. Case Study: Application of MapReduce on Text Classification Using the “20 Newsgroups” Dataset

To demonstrate the application of MapReduce models in practical contexts, we executed a classification experiment on the “20 Newsgroups” dataset [78], a commonly employed corpus of textual data categorized into 20 distinct groups. This dataset is appropriate for evaluating classification tasks inside a MapReduce architecture because of its size and category organization. We implemented a Naive Bayes classifier in a MapReduce simulation, utilizing both single-node and multi-node configurations to evaluate the efficacy of distributed processing.

- The baseline multi-node configuration (utilizing 500 features) was congruent with the single-node metrics, offering a balanced demonstration of MapReduce’s impact in a standard setup.
- The computations adhered to conventional big data performance metrics, rendering the results indicative of MapReduce’s efficacy in processing extensive text datasets.

5.4.1. Dataset Overview

The “20 Newsgroups” collection comprises over 20,000 items categorized into 20 distinct newsgroups, with subjects ranging from “alt.atheism” to “rec.sport.baseball.” Text data from each document was transformed into numerical characteristics utilizing the TF-IDF (Term Frequency-Inverse Document Frequency) approach for our research [79]. This transformation is crucial for facilitating efficient document categorization inside a MapReduce system.

5.4.2. Experiment Setup

The experiments were conducted on a personal computer with the following specifications:

- **Processor:** Intel Core i7-10710U, with six physical cores and 12 logical threads, each operating at a base frequency of 1.1 GHz and turbo frequency of up to 1.6 GHz.
- **Memory (RAM):** 16 GB.
- **Operating System:** Windows with DirectX 12 support.

5.4.3. Performance Metrics Evaluation

The classification model’s performance was evaluated using the metrics outlined in Section 5, which include accuracy, precision, recall, F1 score, Jaccard index, and reaction time. Furthermore, the speedup, characterized as the ratio of single-node to multi-node processing duration, was quantified to demonstrate the advantages of distributed processing.

5.4.4. Results

Comparing the performance of a MapReduce-based classification task performed on a single-node setup versus a multi-node setup with two nodes, both configurations provide key performance metrics, including accuracy, precision, recall, F1 score, Jaccard index, and response time. Furthermore, the table demonstrates the speedup achieved by the multi-node setup, demonstrating the efficiency gains from distributed processing, as shown in Table 5.

Table 5. Performance of a MapReduce-based classification task.

Metric	Single-Node Result	Multi-Node Result	Improvement/Speedup	Number of Nodes
Accuracy	0.672	0.672	N/A	1 (Single), 2 (Multi)
Precision	0.681	0.681	N/A	1 (Single), 2 (Multi)
Recall	0.662	0.662	N/A	1 (Single), 2 (Multi)
F1 Score	0.657	0.657	N/A	1 (Single), 2 (Multi)
Jaccard Index	0.501	0.501	N/A	1 (Single), 2 (Multi)
Response Time (s)	3.191	5.261	Speedup: 0.607 s	1 (Single), 2 (Multi)

Table 5 demonstrates the effects of transitioning from a single-node configuration to a two-node multi-node configuration on classification metrics and processing efficiency in the MapReduce task. The classification metrics—accuracy, precision, recall, F1 score, and Jaccard index—remain unchanged between the single-node and multi-node setups. For instance, accuracy holds steady at 0.672, and F1 score remains at 0.657. This stability indicates that data distribution across nodes in the two-node setup does not degrade classification performance, suggesting that parallelization did not introduce any feature correlation loss or processing inconsistencies in this instance.

However, the impact on processing speed is different. The response time increases from 3.191 s in the single-node configuration to 5.261 s in the two-node setup, resulting in a speedup factor of 0.607 s. This unexpected outcome implies that the overhead associated with managing data distribution and coordination across nodes outweighed the benefits of parallelization in this configuration.

Overall, this analysis highlights a key trade-off in the MapReduce framework. While multi-node configurations are generally intended to improve processing times, the overhead of managing parallel tasks can sometimes negate these benefits, especially in smaller setups or with tasks that may not benefit directly from distributed processing. In this case, the single-node configuration is more efficient for processing speed, while the two-node setup provides similar classification accuracy without an efficiency gain.

6. Research Gaps and Future Scopes

- The Hadoop map-reducing framework faced severe incompatibility issues when processed with diverse nodes, and the security features administered by Hadoop and Linux were not effective. Moreover, the system was not compatible when there were no dedicated nodes or servers for processing [1].
- The CRMAP model failed to mine high-standard sequential structures when evaluated with big data, and the selection of the patterns for mining was not accurate. In addition, the model required updating the patterns when the data were modified or removed [10].
- The MR-MVPP system did not employ a set similarity join approach and utilized a hashing technique, which did not produce accurate results. The coverage rate of the stored views was lower, reducing the analytical queries' responsive time [29].
- The SEWAAN model lacked learning algorithms for evaluating the weights and extracting distinct patterns. The model faced difficulty allocating the appropriate nodes for processing the straggler tasks. Moreover, the large set of input data and the failures reduced accuracy [4].
- The effectiveness of text clustering in the CICC-BDVMR model was reduced since the feature selection and feature reduction process were performed separately. Upgraded optimization strategies, including Harris Hawk optimization and Salp Swarm optimization, were required for improved performance [32].

- The multi-dimensional geospatial mining model necessitated distributed processing approaches for improved clustering performance. The model failed to capture the geospatial data at various levels of speculation due to the lack of a visualization interface and required a workflow pipeline for scheduling the tasks [33].
- The MM-MGSMO model faced difficulties handling massive amounts of complex data against the cloud servers. The utilized swarm-optimized algorithm was ineffective in maintaining privacy; hence, the information could be extracted from the leaked orders. However, the delay in processing the wide range of information was longer [52].
- The MFOB-MP faced vulnerabilities in handling high-dimensional features, reducing the classification accuracy. The FCM algorithm was ineffective with the prototype-based clustering methods, and the model evaluation was not analyzed using various criteria [59].
- The KSC-CMER model was evaluated with a smaller database, yet all the data points available in the dataset were not tested. The standard of the clusters needs to be increased, and the lack of optimal parameters reduced the clustering outcome [46].
- The intensive complications and the fault tolerating factors in the TMaR model were not evaluated. Furthermore, the partition sizes in the intermediate data have to be estimated earlier for improved performance. In addition, the processing time and the energy consumption were higher [40].
- Due to the communication and parallel computing management overhead, multiple MapReduce models are suffering from low speedup [80]. For example, in some cases, the speedup of the MapReduce model is no more than 4 in 48 machines. Low resource utilization significantly limits the scalability of MapReduce models.

Future scopes: Real-time calculations and data stream computation were difficult and could be resolved with in-term processing. Developing an ensemble algorithm could solve the memory dependence issues. Moreover, enhanced optimization was required to minimize load balancing, resource allocation, resource utilization, and fault tolerance challenges. Implementing a cloud-based Hadoop system will improve flexibility, cost-effectiveness, and scalability. This will enable the leveraging of the capabilities of big data without handling intricate infrastructures.

7. Conclusions and Future Direction

Big data impacts various sectors, and managing large-scale distributed data is a challenging and essential research topic. This study covers the various MapReduce models such as Hadoop, Spark, Hive, Pig, Cassandra, and MongoDB for processing big data. This survey delivers an overview of the various map-reducing models and the performance metrics used for the performance evaluation in the research articles. In addition, this review highlights the advantages, achievements, and limitations of improving the field of MapReduce models for processing big data. Further, the survey analyzed different scheduling techniques in MapReduce frameworks as the scheduling influences the performance and efficiency of the system. This survey analyzed the datasets utilized in the research articles with insight into the diverse map-reducing technologies with limitations. In order to provide insight into the MapReduce models' efficiency, metaheuristic optimization algorithms are utilized to optimize the parameters, resulting in improving the performance by minimizing the computational cost, specifically for big data processing and clustering tasks.

Moreover, selecting an appropriate map-reduce method for a particular task is essential for better performance. In addition, the MapReduce models still have limitations, calling for further exploration in processing big data [81,82]. The efforts to address the limitations regarding scalability, error rate, and classification would contribute to the advancement

in processing a wide range of information. Out of all the models, the Hadoop system is predominantly utilized in distinct sectors; however, cluster classification and optimal parameter selection will be promising research directions. Furthermore, Hadoop and Spark can be integrated to enhance the efficiency and processing scale of the algorithm. The map-reduce-based models can be further harnessed in other domains, including insurance, retail, and banking, to aggregate the information from multiple servers.

Author Contributions: Conceptualization, H.B.A. and Y.K.; investigation, H.B.A., Y.Z. and D.T.; methodology, H.B.A. and D.T.; software, H.B.A. and Y.K.; investigation, H.B.A., D.T., Y.Z.; formal analysis, H.B.A. and Y.K.; data curation, H.B.A. and Y.K.; writing—original draft, H.B.A.; writing—review & editing, H.B.A., Y.K., Y.Z. and D.T.; visualization, H.B.A. and Y.K.; supervision, H.B.A.; project administration, H.B.A.; funding acquisition, H.B.A. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Wenzhou-Kean University Internal Research Support Program (IRSPG202202).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All data are based on references.

Acknowledgments: The authors gratefully acknowledge the financial support from Wenzhou-Kean University.

Conflicts of Interest: The authors declare that they have no competing interests.

References

1. Kumar, A.; Varshney, N.; Bhatiya, S.; Singh, K.U. Replication-Based Query Management for Resource Allocation Using Hadoop and MapReduce over Big Data. *Big Data Min. Anal.* **2023**, *6*, 465–477.
2. Ali, I.M.S.; Hariprasad, D. MapReduce Based Hyper Parameter Optimised Extreme Learning Machine For Big Data Classification. *Eur. Chem. Bull.* **2023**, *12*, 7198–7210.
3. Choi, S.Y.; Chung, K. Knowledge process of health big data using MapReduce-based associative mining. *Pers. Ubiquitous Comput.* **2020**, *24*, 571–581.
4. Farhang, M.; Safi-Esfahani, F. Recognizing mapreduce straggler tasks in big data infrastructures using artificial neural networks. *J. Grid Comput.* **2020**, *18*, 879–901. [\[CrossRef\]](#)
5. Peddi, P. An efficient analysis of stocks data using MapReduce. *JASC J. Appl. Sci. Comput.* **2019**, *VI*, 4076–4087.
6. Lin, H.; Su, Z.; Meng, X.; Jin, X.; Wang, Z.; Han, W.; An, H.; Chi, M.; Wu, Z. Combining Hadoop with MPI to Solve Metagenomics Problems that are both Data-and Compute-intensive. *Int. J. Parallel Program.* **2018**, *46*, 762–775.
7. Iwendi, C.; Ponnann, S.; Munirathinam, R.; Srinivasan, K.; Chang, C.Y. An efficient and unique TF/IDF algorithmic model-based data analysis for handling applications with big data streaming. *Electronics* **2019**, *8*, 1331. [\[CrossRef\]](#)
8. Jain, P.; Gyanchandani, M.; Khare, N. Enhanced secured MapReduce layer for big data privacy and security. *J. Big Data* **2019**, *6*, 30.
9. Venkatesh, G.; Arunesh, K. MapReduce for big data processing based on traffic aware partition and aggregation. *Clust. Comput.* **2019**, *22* (Suppl. 5), 12909–12915. [\[CrossRef\]](#)
10. Saleti, S.; Subramanyam, R.B.V. A MapReduce solution for incremental mining of sequential patterns from big data. *Expert Syst. Appl.* **2019**, *133*, 109–125.
11. Asif, M.; Abbas, S.; Khan, M.A.; Fatima, A.; Khan, M.A.; Lee, S.W. MapReduce based intelligent model for intrusion detection using machine learning technique. *J. King Saud Univ.-Comput. Inf. Sci.* **2022**, *34*, 9723–9731.
12. Yange, T.S.; Gambo, I.P.; Ikono, R.; Soriyan, H.A. A multi-nodal implementation of apriori algorithm for big data analytics using MapReduce framework. *Int. J. Appl. Inf. Syst.* **2020**, *12*, 8–28.
13. Chen, W.; Liu, B.; Paik, I.; Li, Z.; Zheng, Z. QoS-aware data placement for MapReduce applications in geo-distributed data centers. *IEEE Trans. Eng. Manag.* **2020**, *68*, 120–136.
14. Vats, S.; Sagar, B.B. An independent time optimized hybrid infrastructure for big data analytics. *Mod. Phys. Lett. B* **2020**, *34*, 2050311.
15. Ramachandran, M.; Patan, R.; Kumar, A.; Hosseini, S.; Gandomi, A.H. Mutual informative MapReduce and minimum quadrangle classification for brain tumor big data. *IEEE Trans. Eng. Manag.* **2021**, *70*, 2644–2655.

16. Alwasel, K.; Calheiros, R.N.; Garg, S.; Buyya, R.; Pathan, M.; Georgakopoulos, D.; Ranjan, R. BigDataSDNSim: A simulator for analyzing big data applications in software-defined cloud data centers. *Softw. Pract. Exp.* **2021**, *51*, 893–920.
17. Kumar, D.; Jha, V.K. An improved query optimization process in big data using ACO-GA algorithm and HDFS MapReduce technique. *Distrib. Parallel Databases* **2021**, *39*, 79–96.
18. Kaur, K.; Garg, S.; Kaddoum, G.; Kumar, N. Energy and SLA-driven MapReduce job scheduling framework for cloud-based cyber-physical systems. *ACM Trans. Internet Technol. (TOIT)* **2021**, *21*, 1–24.
19. Rajendran, S.; Khalaf, O.I.; Alotaibi, Y.; Alghamdi, S. MapReduce-based big data classification model using feature subset selection and hyperparameter tuned deep belief network. *Sci. Rep.* **2021**, *11*, 24138.
20. Banane, M.; Belangour, A. A new system for massive RDF data management using Big Data query languages Pig, Hive, and Spark. *Int. J. Comput. Digit. Syst.* **2020**, *9*, 259–270.
21. Dahiphale, D. Mapreduce for graphs processing: New big data algorithm for 2-edge connected components and future ideas. *IEEE Access* **2023**, *11*, 54986–55001.
22. Verma, N.; Malhotra, D.; Singh, J. Big data analytics for retail industry using MapReduce-Apriori framework. *J. Manag. Anal.* **2020**, *7*, 424–442. [\[CrossRef\]](#)
23. Sardar, T.H.; Ansari, Z. Distributed big data clustering using MapReduce-based fuzzy C-medoids. *J. Inst. Eng. (India) Ser. B* **2022**, *103*, 73–82.
24. Guan, S.; Zhang, C.; Wang, Y.; Liu, W. Hadoop-based secure storage solution for big data in cloud computing environment. *Digit. Commun. Netw.* **2024**, *10*, 227–236.
25. Dang, T.D.; Hoang, D.; Nguyen, D.N. Trust-based scheduling framework for big data processing with MapReduce. *IEEE Trans. Serv. Comput.* **2019**, *15*, 279–293.
26. Jo, J.; Lee, K.W. MapReduce-based D_ELT framework to address the challenges of geospatial Big Data. *ISPRS Int. J. Geo-Inf.* **2019**, *8*, 475. [\[CrossRef\]](#)
27. Meena, K.; Sujatha, J. Reduced time compression in big data using mapreduce approach and hadoop. *J. Med. Syst.* **2019**, *43*, 239.
28. Kulkarni, O.; Jena, S.; Ravi Sankar, V. MapReduce framework based big data clustering using fractional integrated sparse fuzzy C means algorithm. *IET Image Process.* **2020**, *14*, 2719–2727.
29. Azgomi, H.; Sohrabi, M.K. MR-MVPP: A map-reduce-based approach for creating MVPP in data warehouses for big data applications. *Inf. Sci.* **2021**, *570*, 200–224. [\[CrossRef\]](#)
30. Chiang, D.L.; Wang, S.K.; Wang, Y.Y.; Lin, Y.N.; Hsieh, T.Y.; Yang, C.Y.; Shen, V.R.; Ho, H.W. Modeling and analysis of Hadoop MapReduce systems for big data using Petri Nets. *Appl. Artif. Intell.* **2021**, *35*, 80–104. [\[CrossRef\]](#)
31. Abukhodair, F.; Alsaggaf, W.; Jamal, A.T.; Abdel-Khalek, S.; Mansour, R.F. An intelligent metaheuristic binary pigeon optimization-based feature selection and big data classification in a MapReduce environment. *Mathematics* **2021**, *9*, 2627. [\[CrossRef\]](#)
32. Xu, Z. Computational intelligence based sustainable computing with classification model for big data visualization on MapReduce environment. *Discov. Internet Things* **2022**, *2*, 2.
33. Alkathiri, M.; Jhummarwala, A.; Potdar, M.B. Multi-dimensional geospatial data mining in a distributed environment using MapReduce. *J. Big Data* **2019**, *6*, 82. [\[CrossRef\]](#)
34. Gandomi, A.; Reshadi, M.; Movaghar, A.; Khademzadeh, A. HybSMRP: A hybrid scheduling algorithm in Hadoop MapReduce framework. *J. Big Data* **2019**, *6*, 106.
35. Banchhor, C.; Srinivasu, N. Holoentropy based Correlative Naive Bayes classifier and MapReduce model for classifying the big data. *Evol. Intell.* **2022**, *15*, 1037–1050.
36. Gheisari, M.; Wang, G.; Bhuiyan, M.Z.A. A survey on deep learning in big data. In Proceedings of the 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), Guangzhou, China, 21–24 July 2017; Volume 2, pp. 173–180.
37. Ramsingh, J.; Bhuvaneswari, V. An efficient MapReduce-based hybrid NBC-TFIDF algorithm to mine the public sentiment on diabetes mellitus—a big data approach. *J. King Saud Univ.-Comput. Inf. Sci.* **2021**, *33*, 1018–1029. [\[CrossRef\]](#)
38. Xia, D.; Zhang, M.; Yan, X.; Bai, Y.; Zheng, Y.; Li, Y.; Li, H. A distributed WND-LSTM model on MapReduce for short-term traffic flow prediction. *Neural Comput. Appl.* **2021**, *33*, 2393–2410.
39. Tripathi, A.K.; Sharma, K.; Bala, M.; Kumar, A.; Menon, V.G.; Bashir, A.K. A parallel military-dog-based algorithm for clustering big data in cognitive industrial internet of things. *IEEE Trans. Ind. Inform.* **2020**, *17*, 2134–2142. [\[CrossRef\]](#)
40. Maleki, N.; Faragardi, H.R.; Rahmani, A.M.; Conti, M.; Lofstead, J. TMAr: A two-stage MapReduce scheduler for heterogeneous environments. *Hum.-Centric Comput. Inf. Sci.* **2020**, *10*, 1–26.
41. Narayanan, U.; Paul, V.; Joseph, S. A novel system architecture for secure authentication and data sharing in cloud enabled Big Data Environment. *J. King Saud Univ.-Comput. Inf. Sci.* **2022**, *34*, 3121–3135. [\[CrossRef\]](#)
42. Selvi, R.T.; Muthulakshmi, I. Modelling the MapReduce based optimal gradient boosted tree classification algorithm for diabetes mellitus diagnosis system. *J. Ambient Intell. Humaniz. Comput.* **2021**, *12*, 1717–1730.
43. Chawla, T.; Singh, G.; Pilli, E.S. MuSe: A multi-level storage scheme for big RDF data using MapReduce. *J. Big Data* **2021**, *8*, 130.

44. Natesan, P.; Sathishkumar, V.E.; Mathivanan, S.K.; Venkatesan, M.; Jayagopal, P.; Allayear, S.M. A distributed framework for predictive analytics using Big Data and MapReduce parallel programming. *Math. Probl. Eng.* **2023**, *2023*, 6048891.
45. Bhattacharya, N.; Mondal, S.; Khatua, S. A MapReduce-based association rule mining using Hadoop cluster—An application of disease analysis. In *Innovations in Computer Science and Engineering: Proceedings of the Sixth ICICSE 2018*; Springer: Singapore, 2019; pp. 533–541.
46. Maheswari, K.; Ramakrishnan, M. Kernelized Spectral Clustering based Conditional MapReduce function with big data. *Int. J. Comput. Appl.* **2021**, *43*, 601–611. [[CrossRef](#)]
47. Sardar, T.H.; Ansari, Z. MapReduce-based fuzzy C-means algorithm for distributed document clustering. *J. Inst. Eng. (India) Ser. B* **2022**, *103*, 131–142.
48. Krishnaswamy, R.; Subramaniam, K.; Nandini, V.; Vijayalakshmi, K.; Kadry, S.; Nam, Y. Metaheuristic based clustering with deep learning model for big data classification. *Comput. Syst. Sci. Eng.* **2023**, *44*, 391–406.
49. Akhtar, M.N.; Saleh, J.M.; Awais, H.; Bakar, E.A. Map-Reduce based tipping point scheduler for parallel image processing. *Expert Syst. Appl.* **2020**, *139*, 112848. [[CrossRef](#)]
50. Rodrigues, A.P.; Chiplunkar, N.N. A new big data approach for topic classification and sentiment analysis of Twitter data. *Evol. Intell.* **2022**, *15*, 877–887.
51. Madan, S.; Goswami, P. A privacy preservation model for big data in map-reduced framework based on k-anonymisation and swarm-based algorithms. *Int. J. Intell. Eng. Inform.* **2020**, *8*, 38–53. [[CrossRef](#)]
52. Nithyanantham, S.; Singaravel, G. Resource and cost aware glowworm mapreduce optimization based big data processing in geo distributed data center. *Wirel. Pers. Commun.* **2021**, *117*, 2831–2852. [[CrossRef](#)]
53. Liu, J.; Tang, S.; Xu, G.; Ma, C.; Lin, M. A novel configuration tuning method based on feature selection for Hadoop MapReduce. *IEEE Access* **2020**, *8*, 63862–63871. [[CrossRef](#)]
54. Shirvani, M.H. An energy-efficient topology-aware virtual machine placement in cloud datacenters: A multi-objective discrete jaya optimization. *Sustain. Comput. Inform. Syst.* **2023**, *38*, 100856.
55. Mirza, N.M.; Ali, A.; Musa, N.S.; Ishak, M.K. Enhancing Task Management in Apache Spark Through Energy-Efficient Data Segregation and Time-Based Scheduling. *IEEE Access* **2024**, *12*, 105080–105095.
56. Sanaj, M.S.; Prathap, P.J. An efficient approach to the map-reduce framework and genetic algorithm based whale optimization algorithm for task scheduling in cloud computing environment. *Mater. Today Proc.* **2021**, *37*, 3199–3208.
57. Kadkhodaei, H.; Moghadam, A.M.E.; Dehghan, M. Big data classification using heterogeneous ensemble classifiers in Apache Spark based on MapReduce paradigm. *Expert Syst. Appl.* **2021**, *183*, 115369.
58. Usha Lawrance, J.; Nayahi Jesudhasan, J.V. Privacy preserving parallel clustering based anonymization for big data using MapReduce framework. *Appl. Artif. Intell.* **2021**, *35*, 1587–1620.
59. Ravuri, V.; Vasundra, S. Moth-flame optimization-bat optimization: Map-reduce framework for big data clustering using the Moth-flame bat optimization and sparse Fuzzy C-means. *Big Data* **2020**, *8*, 203–217.
60. Li, X.; Xue, F.; Qin, L.; Zhou, K.; Chen, Z.; Ge, Z.; Chen, X.; Song, K. A recursively updated Map-Reduce based PCA for monitoring the time-varying fluorochemical engineering processes with big data. *Chemom. Intell. Lab. Syst.* **2020**, *206*, 104167.
61. Wang, N.; Chen, F.; Yu, B.; Qin, Y. Segmentation of large-scale remotely sensed images on a Spark platform: A strategy for handling massive image tiles with the MapReduce model. *ISPRS J. Photogramm. Remote Sens.* **2020**, *162*, 137–147. [[CrossRef](#)]
62. Tan, X.; Di, L.; Zhong, Y.; Yao, Y.; Sun, Z.; Ali, Y. Spark-based adaptive Mapreduce data processing method for remote sensing imagery. *Int. J. Remote Sens.* **2021**, *42*, 191–207.
63. Salloum, S.; Huang, J.Z.; He, Y. Random sample partition: A distributed data model for big data analysis. *IEEE Trans. Ind. Inform.* **2019**, *15*, 5846–5854.
64. Krishna, K.B.; Nagaseshudu, M.; Kumar, M.K. An Effective Way of Processing Big Data by Using Hierarchically Distributed Data Matrix. *Int. J. Res.* **2019**, *VIII*, 1628–1635.
65. Ramakrishnan, U.; Nachimuthu, N. An Enhanced Memetic Algorithm for Feature Selection in Big Data Analytics with MapReduce. *Intell. Autom. Soft Comput.* **2022**, *31*, 1547–1559. [[CrossRef](#)]
66. Seifhosseini, S.; Shirvani, M.H.; Ramzanpoor, Y. Multi-objective cost-aware bag-of-tasks scheduling optimization model for IoT applications running on heterogeneous fog environment. *Comput. Netw.* **2024**, *240*, 110161. [[CrossRef](#)]
67. Lamrini, L.; Abounaima, M.C.; Talibi Alaoui, M. New distributed-topsis approach for multi-criteria decision-making problems in a big data context. *J. Big Data* **2023**, *10*, 97.
68. Dhamodharavadhani, S.; Rathipriya, R. Region-wise rainfall prediction using mapreduce-based exponential smoothing techniques. In *Advances in Big Data and Cloud Computing: Proceedings of ICBDC18*; Springer: Singapore, 2019; pp. 229–239.
69. Narayana, S.; Chandanapalli, S.B.; Rao, M.S.; Srinivas, K. Ant cat swarm optimization-enabled deep recurrent neural network for big data classification based on MapReduce framework. *Comput. J.* **2022**, *65*, 3167–3180.
70. Kong, F.; Lin, X. The method and application of big data mining for mobile trajectory of taxi based on MapReduce. *Clust. Comput.* **2019**, *22* (Suppl. 5), 11435–11442.

71. Rao, P.S.; Satyanarayana, S. Privacy preserving data publishing based on sensitivity in context of Big Data using Hive. *J. Big Data* **2018**, *5*, 1–20.
72. Brahim, A.A.; Ferhat, R.T.; Zurfluh, G. Model Driven Extraction of NoSQL Databases Schema: Case of MongoDB. In Proceedings of the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2019), Vienna, Austria, 17–19 September 2019; pp. 145–154.
73. Akhtar, M.M.; Shatat, A.S.A.; Al-Hashimi, M.; Zamani, A.S.; Rizwanullah, M.; Mohamed, S.S.I.; Ayub, R. MapReduce with deep learning framework for student health monitoring system using IoT technology for big data. *J. Grid Comput.* **2023**, *21*, 67. [\[CrossRef\]](#)
74. Madan, S.; Goswami, P. k-DDD measure and mapreduce based anonymity model for secured privacy-preserving big data publishing. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.* **2019**, *27*, 177–199. [\[CrossRef\]](#)
75. Zhu, Y.; Samsudin, J.; Kanagavelu, R.; Zhang, W.; Wang, L.; Aye, T.T.; Goh, R.S.M. Fast Recovery MapReduce (FAR-MR) to accelerate failure recovery in big data applications. *J. Supercomput.* **2020**, *76*, 3572–3588. [\[CrossRef\]](#)
76. Banchhor, C.; Srinivasu, N. FCNB: Fuzzy Correlative naive bayes classifier with mapreduce framework for big data classification. *J. Intell. Syst.* **2019**, *29*, 994–1006. [\[CrossRef\]](#)
77. Vennila, V.; Kannan, A.R. Hybrid parallel linguistic fuzzy rules with canopy mapreduce for big data classification in cloud. *Int. J. Fuzzy Syst.* **2019**, *21*, 809–822. [\[CrossRef\]](#)
78. Gupta, Y.K.; Kamboj, S.; Kumar, A. Proportional exploration of stock exchange data corresponding to various sectors using Apache Pig. *Int. J. Adv. Sci. Technol. (IJAST)* **2020**, *29*, 2858–2867.
79. Mitchell, T. Twenty Newsgroups [Dataset]. UCI Machine Learning Repository. 1997. Available online: <https://doi.org/10.24432/C5C323> (accessed on 1 February 2020).
80. Zhao, Y.; Yoshigoe, K.; Xie, M.; Bian, J.; Xiong, K. L-PowerGraph: A lightweight distributed graph-parallel communication mechanism. *J. Supercomput.* **2020**, *76*, 1850–1879. [\[CrossRef\]](#)
81. Zhao, Y.; Yoshigoe, K.; Xie, M.; Zhou, S.; Seker, R.; Bian, J. Lightgraph: Lighten communication in distributed graph-parallel processing. In Proceedings of the 2014 IEEE International Congress on Big Data, Anchorage, AK, USA, 27 June–2 July 2014; pp. 717–724.
82. Zhao, Y.; Yoshigoe, K.; Xie, M.; Zhou, S.; Seker, R.; Bian, J. Evaluation and analysis of distributed graph-parallel processing frameworks. *J. Cyber Secur. Mobil.* **2014**, *3*, 289–316. [\[CrossRef\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.