International Journal of Trend in Scientific Research and Development (IJTSRD)
Volume 9 Issue 3, May-Jun 2025 Available Online: www.ijtsrd.com e-ISSN: 2456 – 6470

MapReduce-based Algorithms for Efficient Big Data Processing

Dr. Gopal Prasad Sharma[1], Prof. Dr. Pawan Kumar Jha[2], Prof. Raj Kumar Thakur[3]

[1]Associate Professor, Purbanchal University School of Science & Technology (PUSAT), Biratnagar, Nepal
[2,3]Professor, Purbanchal University School of Science & Technology (PUSAT), Biratnagar, Nepal

ABSTRACT
MapReduce is a widely used programming model for processing and analyzing large-scale datasets in a distributed computing environment. As the volume of data continues to grow exponentially, MapReduce offers an efficient and scalable solution to manage big data challenges, particularly in areas requiring parallel processing and fault tolerance. This article explores the fundamentals of MapReduce, highlighting its two key phases Map and Reduce they are utilized to process vast amounts of data across distributed systems. Key MapReduce-based algorithms for tasks such as data analysis, sorting, searching, graph processing, and machine learning are discussed in detail, including implementations of the Word Count algorithm, PageRank, k-means clustering, and matrix multiplication. The article further examines the challenges associated with MapReduce, such as inefficiencies in iterative processing and overheads during shuffle and sort phases. It also explores emerging trends and improvements, including the integration of MapReduce with modern frameworks like Apache Spark and its application in cloud computing and AI-driven big data analytics. Finally, the article reflects on the evolving landscape of big data and distributed computing, highlighting the continued relevance and potential of MapReduce in the future of data processing.

KEYWORDS: *Big Data, Data Processing, Distributed Computing, MapReduce, Parallel Processing*

How to cite this paper: Dr. Gopal Prasad Sharma | Prof. Dr. Pawan Kumar Jha | Prof. Raj Kumar Thakur "MapReduce-based Algorithms for Efficient Big Data Processing" Published in International Journal of Trend in Scientific Research and Development (ijtsrd), ISSN: 2456-6470, Volume-9 | Issue-3, June 2025, pp.781-787, URL: www.ijtsrd.com/papers/ijtsrd81126.pdf

IJTSRD81126

I. INTRODUCTION

A. BACKGROUND ON BIG DATA

Today's digital world defines "Big Data" as huge, complex datasets that cannot be processed or analysed using traditional methods [1]. Big data is defined by these five "5Vs": Volume, the massive amounts of data created daily; Velocity, emphasising how rapidly data is generated and handled; While "variety" refers to the wide range of data types, from text and photos to videos and social media posts, "veracity" describes the data's certainty and quality. "Value" emphasises data-driven decision-making. These demonstrate the challenges of processing, analysing, and storing enormous data. Data of this size presents several issues. Traditional approaches can be overwhelmed by real-time data volume and complexity. Scalability, data heterogeneity, and latency without compromising insight accuracy remain issues. Integration of disparate datasets and the requirement to protect sensitive data make innovative processing frameworks essential.

B. INTRODUCTION TO MAPREDUCE

The MapReduce paradigm was created to solve huge dataset processing problems. Google created MapReduce to process and produce huge datasets in parallel [2]. The Map step turns input data into key-value pairs, while the Reduce phase combines the intermediate results to output, both responsibilities are essential to its operation. Breaking tasks into smaller, more manageable chunks, distributing them across multiple nodes, and executing them in parallel improves efficiency and scalability [3]. In big data, MapReduce's ability to manage enormous datasets across platforms is its greatest strength. It optimises resource utilisation, computes complex equations effectively, and automatically re-executes unsuccessful processes to ensure fault tolerance.

MapReduce is essential for data-intensive companies because it allows enormous analysis and insight extraction. This article examines MapReduce-based algorithms and their importance in big data processing. This article examines the design, implementation, and applications of these algorithms to show their ability to solve huge dataset management problems.

II. FUNDAMENTALS OF MAPREDUCE

A. MAPREDUCE

The programming language and processing model MapReduce makes huge data processing easy for distributed systems [4]. It simplifies massive dataset management by removing data distribution, fault tolerance, and parallelisation from application development due to Google. Separate processing of input data into smaller sections generates intermediate key-value pairs during Map. In a word count application, the Map function reads each line and returns a pair of words and their counts (typically started at 1). Sorting and shuffling these intermediate results with the key prepare them for the next stage. Reduce aims to turn studied data into insights. This strategy lets developers focus on logic while the framework handles complex data distribution and task execution. A preconfigured MapReduce application splits, maps, shuffles, reduces, and outputs input data as output [5]. Each operation is executed by a distributed network of nodes, ensuring efficiency and scalability.

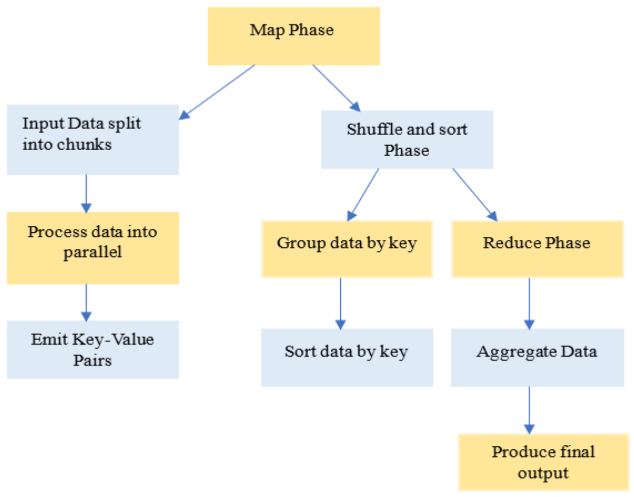

FIGURE 1 MapReduce workflow diagram (Source: Self-Created)

B. KEY FEATURES OF MAPREDUCE

MapReduce is built on three foundational features that make it a robust framework for big data processing:

1. **Scalability**: The framework manages petabytes of data by distributing workload across cluster nodes [6]. It efficiently handles larger datasets or nodes.
2. **Fault Tolerance**: MapReduce supports resilience. The framework automatically reassigns jobs to other nodes if one fails during processing to maintain data integrity [7].
3. **Parallel Processing**: MapReduce reduces calculation time by dividing tasks and processing on multiple nodes.

C. HADOOP AND MAPREDUCE

Hadoop's main processor is MapReduce. Hadoop is an Apache Software Foundation open-source platform [8]. MapReduce applications work well in Hadoop's distributed environment for processing and storing big datasets. The core of Hadoop is Hadoop Distributed File System (HDFS), a fault-tolerant storage system that distributes data across many cluster nodes [9]. HDFS replicates data blocks across several nodes to reduce data loss. MapReduce processes HDFS data blocks and returns the results to the file system. The close relationship between HDFS and MapReduce simplifies distributed data access and processing [10]. Master-slave HDFS is designed, NameNodes control metadata and the file system namespace, whereas DataNodes store data blocks. MapReduce jobs are coordinated by Hadoop's JobTracker or YARN's Resource Manager to optimise task management and resource allocation.

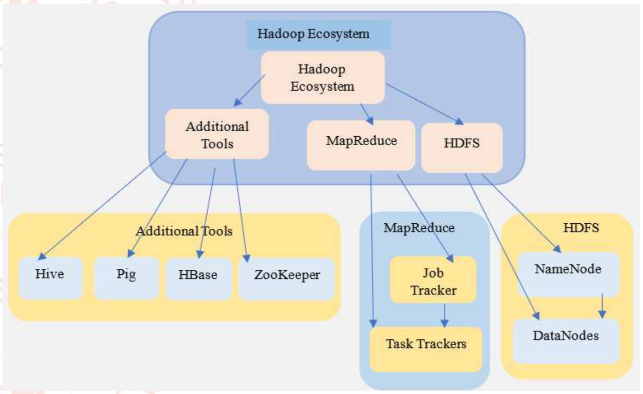

FIGURE 2 Hadoop Ecosystem (Source: Self-Created)

D. ADVANTAGES AND LIMITATIONS OF MAPREDUCE

MapReduce's features make it ideal for processing massive amounts of data [11]. Its concurrent and distributed execution makes it ideal for processing large datasets quickly. Its abstract programming model simplifies distributed systems by shielding developers from their complexities. Despite frequent hardware failures, the fault tolerance system maintains reliability. MapReduce's scalability lets firms handle expanding data volumes without architectural changes. Iterative processing applications like machine learning and graph computations are less efficient since the system must read and write data to disc between iterations [12]. The disc I/O overhead can drastically impact performance. Sometimes the Map and Reduce stages' abstraction is too demanding, limiting its utility for certain calculations. MapReduce is still needed for large data sets, even if Apache Spark can process in-memory.

Study about MapReduce's workflow, foundations, and relationship with Hadoop to understand its limitations and appreciate its role in turning huge data into useful insights.

III. MAPREDUCE-BASED ALGORITHMS

A. OVERVIEW OF MAPREDUCE ALGORITHMS

MapReduce algorithms are designed to efficiently process big datasets using distributed and parallel processing [13]. These algorithms meet essential MapReduce criteria. A network of nodes can do smaller, more manageable jobs concurrently by dividing the problem. The Map stage converts input data into key-value pairs. The Reduce phase aggregates data and these algorithms can redistribute work after errors, making them scalable and fault-tolerant and ensuring constant execution.

B. ALGORITHMS FOR DATA ANALYSIS

1. WORD COUNT ALGORITHM

A popular MapReduce method for huge datasets is Word Count. It shows that parallel processing works by breaking jobs into smaller pieces and assigning them to processors in different locations. Cartography, Sorting and Shuffle, and Reducing are key.

1. **Map Phase**: This stage splits text file input into lines or chunks. We then divided each line's words. Each word has a key-value pair with the word as the key and 1 as the value for a single occurrence.
2. **Shuffle and Sort**: The intermediate key-value pairs assigned to each cluster node change randomly. Key terms match identical words. The subsequent reduction step treats all term instances simultaneously.
3. **Reduce Phase**: The algorithm concludes by adding word counts. After adding all counts, the reducer gets the dataset word frequency.

2. PAGERANK ALGORITHM

The Google-created PageRank algorithm ranks websites by the number and quality of links linking to them. The algorithm shows MapReduce's huge distributed computation capabilities. MapReduce is an excellent way to distribute and parallelise such a big computation, while PageRank iterates.

1. **Map Phase**: During Map phase, each web page communicates its rank to all its connected pages [14]. Every page sends a value to its linked sites depending on its rank split by its link count. The rank value distribution of each page shows its importance.
2. **Shuffle and Sort**: We compile and organise all pages that affect a web page's rank at this step. The rank contributions are grouped by the target page, which receives links.
3. **Reduce Phase**: The Reduce step adds all contributor ranks. Because each link contributes differently, a damping factor is used. Every page gets a new ranking based on its links' relevancy.

C. ALGORITHMS FOR DATA SORTING AND SEARCHING

1. SORT-MERGE AND EXTERNAL SORTING

Massive data processing uses data sorting to aggregate, search, and query data. MapReduce efficiently sorts big datasets when data cannot be stored in memory. MapReduce sorting techniques like Sort-Merge handle massive data sets across several nodes [15]. Mapping begins with mapper nodes locally processing each input piece. Data is organised into key-value pairs with sorting-related values or identifiers as keys and data as values. Before outputting intermediate key-value pairs, mappers sort chunks locally.

Intermediate key-value pairs are sorted and shuffled after Map. Shuffle consolidates and sends reducers key-shared data. To sort records globally, MapReduce provides key-value pairs to reducers. Sorting data throughout the distributed system is crucial for large datasets and further reduce process data. Sorted key-value pairs are output by the reducer. After the shuffle, the dataset is fully sorted by key, making merging easy. Data is sorted outside RAM using disc storage and MapReduce's distributed capabilities.

2. GREP: SEARCHING FOR PATTERNS

MapReduce uses the robust Grep algorithm to explore large text-based datasets for patterns or regular expressions [16]. This method helps with log analysis, text processing, and data mining. These include examining massive unstructured data for critical information. In MapReduce, Grep divides pattern matching into smaller, parallelizable tasks for efficient and scalable data processing.

In Map, each dataset line is handled separately. Mappers match search parameters to pre-created patterns to find lines. One key-value pair is returned when a line matches the pattern. Keys are lines or identities, and values are usually 1 to indicate matches. This helps the program locate the dataset pattern. Map intermediate key-value pairs are reduced. The map step discovers matched lines, so the reducer may output them, simplifying the reduce process. Parallelism speeds up massive dataset searches that single-node algorithms cannot handle. Log analysis intensively searches system or event logs for issue signals or patterns.

D. GRAPH PROCESSING ALGORITHMS

1. BREADTH-FIRST SEARCH (BFS)

Breadth-First Search (BFS) finds the shortest path between any two network nodes. MapReduce helps implement BFS quickly on large graphs that surpass a machine's memory [17]. MapReduce's distributed nature makes BFS valuable for social network analysis, route planning, and web search indexing. In the Map phase, all network nodes broadcast neighbour data. All network nodes notify neighbours of source node distance. Node distance rises by one without a visit. Next-to-next-node keys and updated distance values are in key-value pairs. Thus, the graph may be examined simultaneously with each node sending its distance to neighbours.

The Shuffle and Sort phase of MapReduce sorts key-value pairs by node (key) to assign each update to the same reducer. This stage manages the distributed BFS process and assures node distance aggregation and reduce averaged node distances. BFS reducers choose the shortest (or first-discovered) distance for each node. Printing updated distances for the next generation. After visiting all reachable nodes and identifying the shortest path from the source to all others, the method propagates graph distance.

2. CONNECTED COMPONENTS ALGORITHM

The Connected Components algorithm groups graph nodes with paths between every pair of nodes.

Social network research, bioinformatics, and other fields that require to uncover clusters or subnetworks use this graph analysis method [18]. Dividing a graph into its connected components helps us uncover communities in social networks and functional modules in biological networks.

Every node broadcasts its own and its neighbours' identifiers during the Map phase. These emissions allow MapReduce to communicate with each node independently. Mapper output key-value pairs use the node identifier as key and the neighbour list as value. Every node must send its connection information to MapReduce to consolidate links in the next step. In the Reduce step, the system arranges key-value pairs by node identifiers. The reducer merges all neighbour lists of nodes associated to a key (node) to form a single connected component. Next, the reducer assigns the same ID to all component nodes. Labelling all nodes with the same identity easily finds related graph components.

E. MACHINE LEARNING WITH MAPREDUCE

1. K-MEANS CLUSTERING

One of the most common unsupervised machine learning methods, K-means clusters data points by centroids. Each data point receives the nearest centroid first, then iteratively. The new points are used to recalculate centroids. K-means may be parallelised and scaled over distributed computers with MapReduce, making it excellent for large datasets [19]. Map phase assigns data points to closer cluster centroid. The mapper outputs the data point and cluster ID (the closest centroid) as a key-value pair. The mapper processes each data point separately and clusters points closest to the centroid. This parallelisation ensures the approach can handle large datasets by letting each node process a subset of data points.

During the Reduce phase, the system aggregates key-value pairs by cluster ID (key) to send all points to the same reducer. The reducer averages the locations of all assigned points to recalculate each cluster's centre. Next algorithm iteration uses updated centroids and outputs. As the process continues, centroids are refined based on their data points. Iterates until the centroids are stable, with few or no position change. K-means is ideal for big data applications like customer segmentation, photo analysis, and anomaly detection in large datasets since MapReduce's scalability allows computing to be distributed across many servers. K-means can efficiently analyse enormous data volumes on MapReduce, enabling distributed clustering.

2. LINEAR REGRESSION

Linear regression, an essential machine learning tool, models the relationship between independent variables (x) and dependent variables (y). Many industries, including engineering and economics, use it to predict values and analyse variables. MapReduce scales linear regression operations across multiple nodes in a distributed context, ensuring speed and scalability for large datasets.

1. **Map Phase**: In the Map phase, each data point contributes to the computation of intermediate sums that are required for the regression analysis. Specifically, for each data point (x, y), the mapper calculates the following components:

- xxx (the independent variable),
- yyy (the dependent variable),
- x×yx times yx×y (the product of the independent and dependent variables), and
- x2x^2x2 (the square of the independent variable).

The mapper emits these values as key-value pairs where the key is a common placeholder (e.g., null or 1), and the values are the individual computed sums for each data point. This allows the intermediate results to be grouped and processed in parallel across many machines in the MapReduce framework.

2. **Reduce Phase**: In the Reduce phase, the system groups the intermediate results by their key and aggregates the sums of xxx, yyy, x×yx \times yx×y, and x2x^2x2. The reducer combines all the values and computes the final sums that are required to calculate the regression coefficients (i.e., slope and intercept) of the linear regression model. The

$$\text{Slope } \beta_1 = \frac{n\sum(xy) - \sum x \sum y}{n\sum x^2 - (\sum x)^2}$$

$$\text{Intercept } \beta_0 = \frac{\sum y - \beta_1 \sum y}{n}$$

In this method, each node handles a subset of data points and then combines their findings in the Reduce phase to solve large-scale regression problems in parallel. MapReduce spreads processing, ensuring regression analysis speed and scalability.

F. OPTIMIZED ALGORITHMS

Matrix multiplication is vital in scientific computing, computer graphics, and machine learning. Though designed for parallelizable jobs, MapReduce algorithms' iterative nature makes them inefficient for this problem. Efficient and scalable MapReduce matrix multiplication solutions offer targeted ways to tackle these challenges with huge datasets [20]. In the Map phase, the elements of the two matrices are assigned IDs, usually their row and column indices. To multiply two matrices, these element identifiers are needed. For example, if we are multiplying Matrix A (of size m×nm \times nm×n) with Matrix B (of size n×pn \times pn×p), the mapper will emit key-value pairs where the key represents a pair of indices from the respective matrices (such as (i, k) for Matrix A and (k, j) for Matrix B). The value for these key-value pairs will be the matrix elements themselves (A[i][k] and B[k][j]).

In the Reduce phase, matching elements from the two matrices are grouped based on their common key (i.e., the row-column indices). For each key (i, j), the reducer multiplies the corresponding elements from Matrix A and Matrix B (i.e., A[i][k]×B[k][j]A[i][k] \times B[k][j]A[i][k]×B[k][j]) and sums them over all the possible k values to compute the resulting element C[i][j]C[i][j]C[i][j] in the final product matrix. This process effectively computes the elements of the resulting matrix CCC. Since matrix multiplication involves several steps, MapReduce must optimise this technique. To expedite computations and eliminate disc I/O overhead, in-memory caches are often used.

Optimised matrix partitioning techniques simplify data shuffling and sorting between map and reduce phases. Reduced overheads make MapReduce-based matrix multiplication cheaper for massively parallel data processing applications like ML model training and physical system simulation.

MapReduce is ideal for HPC with big data because iterative optimisation is ideal for matrix multiplication and its meticulous design lets it manage large matrices that don't fit in memory.

IV. APPLICATIONS OF MAPREDUCE IN BIG DATA PROCESSING

A. INDUSTRY APPLICATIONS

1. E-COMMERCE

MapReduce is essential for online shops' log analysis and recommendation systems. E-commerce platforms monitor user interactions, buying habits, and website traffic [21]. These logs can be processed using MapReduce to provide most frequented sites, peak activity hours, and behavioural trends to improve user experience and website speed. Recommendation systems, which analyse user preferences and purchases, are another important MapReduce application. MapReduce executes collaborative filtering algorithms to determine which products users will buy to increase customisation and sales.

2. HEALTHCARE

Healthcare uses MapReduce to evaluate patient data and detect disease patterns. Medical imaging, genetic, and EHR data can be processed using MapReduce to uncover trends, predict sickness outbreaks, and improve treatment outcomes [22]. MapReduce is used in DNA sequencing. This discipline searches billions of nucleotide sequences for mutations and disease indicators.

3. FINANCE

Financial uses of MapReduce include fraud detection and real-time transaction analysis. MapReduce is used by fraud detection systems to evaluate transaction data for anomalies like unusual spending patterns or high-frequency transactions [23]. These tools alert investigators to fraud. Real-time transaction analysis is another major usage. MapReduce, which processes transaction data, helps financial institutions quickly identify trends, assess risk, and make decisions.

B. USE CASES

1. SOCIAL MEDIA ANALYTICS

Social media platforms generate massive amounts of unstructured data, including multimedia, comments, likes, shares, and posts [24]. Processing this data for insights requires MapReduce. Sentiment analysis is used to assess public opinion on a brand, product, or event by analysing user-generated information. MapReduce algorithms process hashtags, keywords, and other metadata to detect sentiment trends.

2. SCIENTIFIC COMPUTING

MapReduce has revolutionised scientific computing by simplifying complex dataset processing [25]. This technology helps astronomy, bioinformatics, and climate modelling sort through extensive theoretical and empirical data.

Climate models employ MapReduce to analyse gigabytes of satellite data to predict weather, natural disasters, and climate change.

V. CHALLENGES AND FUTURE DIRECTIONS

A. CHALLENGES IN USING MAPREDUCE FOR BIG DATA

MapReduce struggles with iterative processes in machine learning, graph processing, and other analytical tasks. This I/O cost in MapReduce iterations slows performance because receiving data from disc, processing it, and sending it back is time consuming. Traditional MapReduce cannot be utilised for k-means clustering or PageRank since they require several rounds to converge.

Although essential to MapReduce, sort and shuffle can introduce significant overheads, especially for large datasets. These stages require sorting intermediate data by keys and redistributing it amongst nodes, which is computationally and network expensive. Large datasets can reduce framework efficiency because these operations take time and resources.

B. EMERGING TRENDS AND IMPROVEMENTS

Integrating MapReduce with modern big data frameworks like Apache Spark is becoming more frequent to overcome its limitations. Spark reduces iterative calculation input/output overhead using in-memory MapReduce. Spark is preferred for repetitive calculations because it stores data in memory between iterations, improving speed.

Distributed computing advances have optimised algorithms and data structures for MapReduce systems. Frameworks offer data division and indexing, which reduce communication costs by limiting data movement and maximising locality. Speculative execution and adaptive task scheduling improve fault tolerance and resource use.

C. FUTURE DIRECTIONS

Cloud computing has given MapReduce new opportunities, especially for elastic and scalable large data solutions. Managed MapReduce services from Amazon Web Services (AWS) and Google Cloud let companies handle huge datasets without installing infrastructure. Future advances may optimise MapReduce for serverless architectures, where resources are dynamically assigned based on workload demands for cost-effectiveness and scalability. Using MapReduce with AI and machine learning is another potential approach. MapReduce preprocesses huge datasets for machine learning model training using distributed computing. MapReduce is used for clustering, classification, and collaborative filtering in Apache Mahout. MapReduce might be used with TensorFlow or PyTorch to enable distributed training of complicated neural networks on enormous datasets.

VI. CONCLUSION

MapReduce, an early and effective framework for managing distributed large-scale datasets, revolutionised big data processing. Organisations dealing with growing data volumes need it because its Map and Reduce phases simplify complex procedures. Due to its scalability, fault tolerance, and parallel processing, MapReduce can handle enormous data storage, computing, analysis, and visualisation. The MapReduce-based algorithms in this article have helped data analysis, sorting, searching, graph processing, and machine learning. Word Count and PageRank show how MapReduce efficiently processes enormous volumes of unstructured data for web crawling and real-time analytics.

Complex applications like k-means clustering and matrix multiplication demonstrate MapReduce's adaptability to scientific computing and machine learning's iterative needs.

MapReduce may thrive in the ever-changing big data world due to cloud computing, AI integration, and real-time data processing. Apache Spark and hybrid frameworks are improving MapReduce's iterative processing and shuffling overheads, although scalability and fault tolerance remain crucial. MapReduce's flexibility to adapt to new technology will aid big data and distributed computing. MapReduce is vital for processing huge data since it is efficient, scalable, and fault-tolerant.

REFERENCE

[1] P. Kijsanayothin, G. Chalumporn, and R. Hewett, "On using MapReduce to scale algorithms for Big Data analytics: a case study," Journal of Big Data, vol. 6, pp. 1–20, 2019.

[2] L. Abualigah and B. A. Masri, "Advances in MapReduce big data processing: platform, tools, and algorithms," in Artificial Intelligence and IoT: Smart Convergence for Eco-Friendly Topography, pp. 105–128, 2021.

[3] M. Khader and G. Al-Naymat, "Density-based algorithms for big data clustering using

MapReduce framework: A Comprehensive Study," ACM Computing Surveys (CSUR), vol. 53, no. 5, pp. 1–38, 2020.

[4] N. Zhang, M. Wang, Z. Duan, and C. Tian, "Verifying properties of MapReduce-based big data processing," IEEE Trans. Reliab., vol. 71, no. 1, pp. 321–338, 2020.

[5] S. Y. Choi and K. Chung, "Knowledge process of health big data using MapReduce-based associative mining," Pers. Ubiquitous Comput., vol. 24, pp. 571–581, 2020.

[6] S. Heidari, M. Alborzi, R. Radfar, M. A. Afsharkazemi, and A. Rajabzadeh Ghatari, "Big data clustering with varied density based on MapReduce," Journal of Big Data, vol. 6, no. 1, p. 77, 2019.

[7] F. Qi, "A MapReduce-based approach to social network big data mining," J. Comput. Methods Sci. Eng., (Preprint), pp. 1–13, 2023.

[8] R. Tekieh and Z. Beheshti, "A MapReduce-based big data clustering using swarm-inspired meta-heuristic algorithms," Scientia Iranica, 2024.

[9] X. Tan, L. Di, Y. Zhong, Y. Yao, Z. Sun, and Y. Ali, "Spark-based adaptive MapReduce data processing method for remote sensing imagery," Int. J. Remote Sens., vol. 42, no. 1, pp. 191–207, 2021.

[10] I. A. T. Hashem et al., "MapReduce scheduling algorithms: a review," J. Supercomput., vol. 76, pp. 4915–4945, 2020.

[11] T. H. Sardar and Z. Ansari, "Distributed big data clustering using MapReduce-based fuzzy C-medoids," J. Inst. Eng. India Ser. B, vol. 103, no. 1, pp. 73–82, 2022.

[12] Y. Mao et al., "A MapReduce-based K-means clustering algorithm," J. Supercomput., pp. 1–22, 2022.

[13] P. Wei, F. He, L. Li, C. Shang, and J. Li, "Research on large data set clustering method based on MapReduce," Neural Comput. Appl., vol. 32, pp. 93–99, 2020.

[14] L. Luo, "Design of big data algorithm based on MapReduce," in Proc. 2020 Int. Conf. Aviation Safety Inf. Technol., pp. 722–724, Oct. 2020.

[15] M. Asif et al., "MapReduce based intelligent model for intrusion detection using machine learning technique," J. King Saud Univ. Comput. Inf. Sci., vol. 34, no. 10, pp. 9723–9731, 2022.

[16] M. Q. Bashabsheh, L. Abualigah, and M. Alshinwan, "Big data analysis using hybrid meta-heuristic optimization algorithm and MapReduce framework," in Integrating Meta-Heuristics and Machine Learning for Real-World Optimization Problems, pp. 181–223, Cham: Springer, 2022.

[17] M. R. Sundara Kumar and H. S. Mohan, "Improving big data analytics data processing speed through MapReduce scheduling and replica placement with HDFS using genetic optimization techniques," J. Intell. Fuzzy Syst., (Preprint), pp. 1–20, 2024.

[18] M. R. Sundarakumar, G. Mahadevan, R. Somula, S. Sennan, and B. S. Rawal, "An approach in big data analytics to improve the velocity of unstructured data using MapReduce," Int. J. Syst. Dyn. Appl., vol. 10, no. 4, pp. 1–25, 2021.

[19] H. Jeong and K. J. Cha, "An efficient MapReduce-based parallel processing framework for user-based collaborative filtering," Symmetry, vol. 11, no. 6, p. 748, 2019.

[20] A. Saxena, A. Chaurasia, N. Kaushik, and N. Kaushik, "Handling big data using MapReduce over hybrid cloud," in Proc. Int. Conf. Innovative Comput. Commun. (ICICC) 2018, vol. 2, pp. 135–144, 2019.

[21] C. Banchhor and N. Srinivasu, "Analysis of Bayesian optimization algorithms for big data classification based on MapReduce framework," Journal of Big Data, vol. 8, no. 1, p. 81, 2021.

[22] T. H. Sardar and Z. Ansari, "An analysis of distributed document clustering using MapReduce based K-means algorithm," J. Inst. Eng. India Ser. B, vol. 101, no. 6, pp. 641–650, 2020.

[23] E. Gothai et al., "MapReduce based distance weighted k-nearest neighbor machine learning algorithm for big data applications," Scalable Comput. Pract. Exp., vol. 23, no. 4, pp. 129–145, 2022.

[24] T. H. Sardar and Z. Ansari, "MapReduce-based fuzzy C-means algorithm for distributed document clustering," J. Inst. Eng. India Ser. B, vol. 103, no. 1, pp. 131–142, 2022.

[25] C. M. Chao, P. Z. Chen, S. Y. Yang, and C. H. Yen, "An efficient MapReduce-based apriori-like algorithm for mining frequent itemsets from big data," in Proc. 11th EAI Int. Conf. Wireless Internet (WiCON) 2018, Taipei, Taiwan, Oct. 2018, pp. 76–85.