Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2024 SCPE. Volume 25, Issues 6, pp. 4558–4575, DOI 10.12694/scpe.v25i6.3294

OPTIMIZING HADOOP DATA LOCALITY: PERFORMANCE ENHANCEMENT STRATEGIES IN HETEROGENEOUS COMPUTING ENVIRONMENTS

SI-YEONG KIM[*] AND TAI-HOON KIM[†]

Abstract. As organizations increasingly harness big data for analytics and decision-making, the efficient processing of massive datasets becomes paramount. Hadoop, a widely adopted distributed computing framework, excels in processing large-scale data. However, its performance is contingent on effective data locality, which becomes challenging in heterogeneous computing environments comprising diverse hardware resources. This research addresses the imperative of enhancing Hadoop's data locality performance in heterogeneous computing environments. The study explores strategies to optimize data placement and task scheduling, considering the diverse characteristics of nodes within the infrastructure. Through a comprehensive analysis of Hadoop's data locality algorithms and their impact on performance, this work proposes novel approaches to mitigate challenges associated with disparate hardware capabilities. Weighted Extreme Learning Machine Technique (Weighted ELM) with the Firefly Algorithm (WELM-FF) is used in the proposed work. The integration of Weighted Extreme Learning Machine (WELM) with the Firefly Algorithm holds promise for enhancing machine learning models in the context of large-scale data processing. The research employs a combination of theoretical analysis and practical experiments to evaluate the effectiveness of the proposed enhancements. Factors such as network latency, disk I/O, and CPU capabilities are taken into account to develop a holistic framework for improving data locality and, consequently, overall Hadoop performance. The findings presented in this study contribute valuable insights to the field of distributed computing, offering practical recommendations for organizations seeking to maximize the efficiency of their Hadoop deployments in heterogeneous computing environments. By addressing the intricacies of data locality, this research strives to enhance the scalability and performance of Hadoop clusters, thereby facilitating more effective utilization of big data resources.

Key words: Hadoop; Data Locality; Performance Enhancement; Heterogeneous Computing; Distributed Computing; Big Data.

1. Introduction.

In the era of big data, the efficient processing and analysis of massive datasets have become critical for organizations across various domains. Hadoop, a widely adopted distributed computing framework, has emerged as a powerful tool for handling large-scale data processing tasks. One of the key factors influencing Hadoop's performance is the effective utilization of data locality, ensuring that computation occurs close to where the data resides as shown in Figure 1.1. However, in heterogeneous computing environments encompassing diverse hardware resources, optimizing data locality presents significant challenges. This research focuses on addressing these challenges and enhancing the performance of Hadoop in such environments. By exploring strategies to optimize data placement and task scheduling, considering the distinct characteristics of nodes within the infrastructure, this study aims to provide practical insights for organizations seeking to maximize the efficiency of their Hadoop deployments in diverse computing environments [1].

The proliferation of heterogeneous computing environments, comprising a mix of hardware architectures and capabilities, adds a layer of complexity to the already intricate landscape of big data processing. The diverse performance characteristics of nodes within such environments impact data processing efficiency, making it imperative to devise strategies that not only adapt to this heterogeneity but also enhance Hadoop's data locality performance. This research delves into the intricacies of Hadoop's data [2] locality algorithms, scrutinizing their efficacy in heterogeneous settings. By addressing challenges related to network latency, disk I/O, and varying CPU capabilities, the study aims to propose novel approaches for optimizing data placement and task scheduling. Through a blend of theoretical analysis and practical experiments, the research seeks to contribute valuable insights that can guide organizations in tailoring their Hadoop deployments for optimal

[*]School of Electrical and Computer Engineering, Yeosu Campus, Chonnam National University, 59626, Republic of Korea (siyeong23@jnu.ac.kr)
[†]School of Electrical and Computer Engineering, Yeosu Campus, Chonnam National University, 59626, Republic of Korea (Corresponding Author, taihoonn@chonnam.ac.kr)

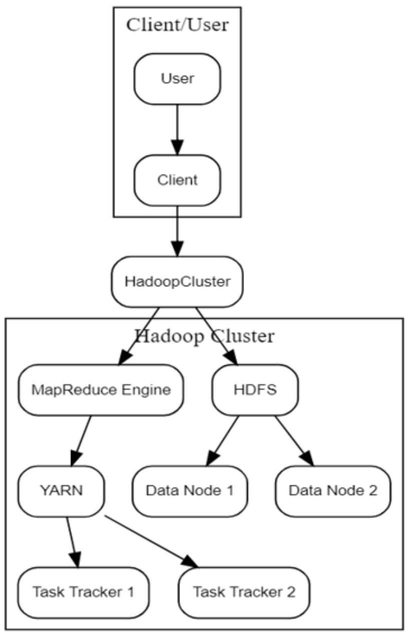

Fig. 1.1: Hadoop Architecture

performance [3] in heterogeneous computing environments. Ultimately, the goal is to empower enterprises to extract maximum value from their big data resources while navigating the complexities introduced by diverse hardware configurations.

In the vast landscape of distributed databases, characterized by a superabundance of data from diverse fields, managing the manifold, intricate, dissimilar, and autonomous nature of occurrences poses a significant challenge [4]. The sheer volume of data, often rife with excessive and extraneous features, necessitates the application of a suitable filtering model to address noise in features or occurrences. Effectively dealing with this substantial number of characteristics requires the implementation of an efficient attribute selection model to choose ranked features for classification or clustering.

A notable contribution by [5] involved the application of a concise set of rules using a feature selection model within rough set theory. Despite these advancements, the classification algorithms encounter a crucial challenge in the form of error rates and class imbalances.

Handling high-dimensional document spaces, especially in the context of large document sets, presents inherent difficulties in preprocessing and classifying. To enhance the learning of classification algorithms, a considerable number of samples [6] need to be learned based on their dimensions. Conceptually, this document space can be viewed as a low-dimensional sub-space enveloped within an ambient space. In response to the dimensionality issue, numerous dimension reduction methods have been developed, aiming to decrease document dimensions and improve performance and efficiency.

The application of these methods becomes particularly crucial in addressing the challenges posed by online medical databases. Arithmetic computing procedures, data accessing, semantics, and domain knowledge are fundamental challenges in dealing [7] with these databases, exacerbated by the complexities, continuous fluctuations, and noise inherent in the growing data landscape. As research endeavors continue, finding effective solutions to these challenges becomes paramount for leveraging the full potential of online medical databases in various applications [8].

The aim of the proposed work is to address the challenges associated with the superabundance of data in distributed databases [9], particularly focusing on the manifold, intricate, dissimilar, and autonomous nature of occurrences across diverse fields. The primary goal is to develop and implement effective filtering models to handle noisy features and occurrences within the substantial dataset [10]. Additionally, the research aims to tackle the issues related to high-dimensional document spaces, where preprocessing and classifying large document sets prove to be challenging.

To achieve these objectives, the proposed work seeks to employ advanced attribute selection models, inspired by rough set theory and other relevant methodologies, to effectively manage the extensive characteristics inherent in the data [11]. The utilization of dimension reduction methods is a key aspect of the research, with the aim of enhancing the learning process of classification algorithms by decreasing document dimensions and improving overall performance and efficiency. Furthermore, the proposed work aims to contribute to the field by addressing challenges in online medical databases, focusing on arithmetic computing procedures, data accessing, semantics, and domain knowledge. The ultimate objective is to develop solutions that navigate the complexities, continuous fluctuations, and noise within the growing data of medical databases, ensuring their effective utilization in various applications.

The proposed work aims to advance the understanding and methodologies for managing vast and complex datasets, with a specific focus on distributed databases and high-dimensional document spaces. The overarching goal is to contribute to the development of robust models and techniques that can enhance the efficiency, accuracy, and applicability of classification algorithms and data processing in challenging environments.

The integration of Weighted Extreme Learning Machine (Weighted ELM) with the Firefly Algorithm presents a potent approach for optimizing machine learning models efficiently. Weighted ELM, an extension of the Extreme Learning Machine algorithm, is known for its computational efficiency and rapid training. By incorporating the Firefly Algorithm, a nature-inspired optimization technique, the model's parameters can be fine-tuned to enhance generalization performance. This synergy benefits from the Firefly Algorithm's ability to efficiently explore and converge to optimal solutions in complex optimization spaces. When applied in the context of Hadoop, a distributed computing framework, the proposed work emphasizes the crucial aspect of data locality. Leveraging Hadoop's parallel processing capabilities, the distributed nature of the Weighted ELM with Firefly Algorithm ensures that computation occurs in close proximity to the data, minimizing data transfer across the nodes of the Hadoop cluster. This strategic use of data locality optimizes the training and optimization processes, facilitating the effective handling of large-scale datasets in a distributed computing environment.

The outline of the paper looks like this: Previous studies are reviewed in Section 2, the methodology for the current study is laid out in Section 3, the experimental data and their analysis are presented and discussed in Section 4, and finally, ideas for future studies are provided in Section 5.

2. Literature Survey. Numerous studies have explored the utility of Data Placement (DP) in the context of MapReduce jobs within Hadoop clusters, addressing challenges in managing distributed databases and high-dimensional document spaces. Noteworthy contributions include a solution to duplicate data files during staged predictive inspection, employing a probability hypoproposed work for replication. Al-Khateeb and Masud introduced a method using rough set theory, enhancing both convenience and safety compared to conventional approaches [12].

The Data-Grouping-AWare (DRAW) technology, developed by another group, focuses on increasing the effectiveness of data-intensive applications in concentrated user interest areas. DRAW optimizes information retrieval from framework log data, achieving maximal parallelism without compromising load balancing. Experimental results demonstrate improved throughput for local map tasks, enhancing overall execution performance compared to Hadoop's default random placement [13].

The Snakelike Data Placement method (SLDP) introduces a technique considering heterogeneity, redistributing information blocks across nodes while addressing data heat. SLDP shows potential to increase MapReduce performance, utilizing less space and energy according to experiments with real-world datasets [14].

Dynamic Replica Strategy (DRS) and Hierarchical Replica Placement Strategy (HRPS) in Hadoop Distributed File System (HDFS) ensure data integrity and accessibility. The Markov model informs the development of a Secure HDFS (Sahds) that adapts the SFAS section designation mechanism. Sahds distributes file contents over similar nodes, maintaining replications and addressing data replication issues [15, 16].

Hybrid mechanisms in HDFS incorporate Solid-State Disks and conventional hard discs to enhance per-

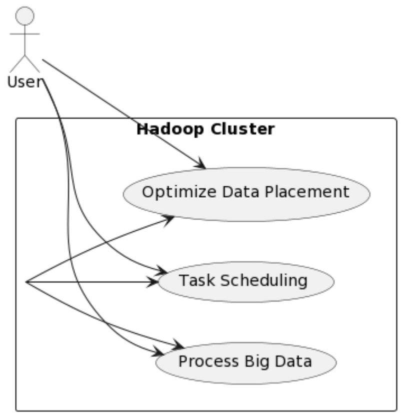

Fig. 3.1: Map Reduce Framwork with Hadoop Cluster

formance without increased energy usage, showing a significant reduction in energy consumption compared to hybrid configurations [17].

A security-aware information placement method for scientific operations in the cloud is proposed, guaranteeing data privacy, integrity, and authentication access. The method calculates the cost of data center security services using a security model and dynamically selects the best data centers with an Ant Colony Optimization (ACO) based approach [18].

Hybrid clouds present challenges in retrieving data while protecting user anonymity. Last-HDFS addresses data placement issues, enabling location-aware cloud storage and file loading based on policy compliance [19].

Clustering methods for organizing data, including K-means and bio-inspired algorithms, have been explored for their simplicity and optimization capabilities. Various bio-inspired algorithms, such as Genetic Algorithm (GA), Differential Evolution (DE), and Particle Swarm Optimization (PSO), address optimization challenges in clustering. A novel approach using GA clustering, DE, and ACO demonstrates potential improvements in cluster centroids' search capability [20].

MapReduce-based methods, including K-means parallelization and parallel K-PSO, are proposed for large-scale clustering, addressing challenges with PSO algorithm scalability in very big datasets. Scalable MR-CPSO uses the parallel MapReduce approach to overcome PSO algorithm issues with large datasets.

In conclusion, these studies collectively contribute to the advancement of distributed data processing, data placement strategies, and clustering methods, with a particular emphasis on enhancing performance, energy efficiency, and security in diverse computing environments.

3. Proposed Methodology. Hadoop is a free, Java-dependent programming framework which supports the processing of massive databases in a distributed computing platform. In addition to efficiency and management of big data, the Hadoop framework offers numerous features such as high availability (recreation of data squares across nodes), adaptation to internal failure (the system itself is built to differentiate and cope with setbacks in the application layer), etc. It is adopted for processing and storing colossal data in distributed environments using programming models. Hadoop appears as the persuasive solution for big data issues with some additional capabilities. Hadoop uses the MapReduce algorithm to run applications. MapReduce is a framework employed to process the enormous databases parallely. It performs distributed operational programming as shown in Figure 3.1.

The MapReduce framework allows to specify information transformation logic through exploiting their own map function and reduce function. MapReduce technology also enhances the monitoring and scheduling of tasks and simplifies massive computing and handling of colossal data volumes. MapReduce being Hadoop's core component, processes massive information in parallel through splitting the job into a cluster of separate tasks. As MapReduce operates in a data-intensive environment, the important factor that directly affects the

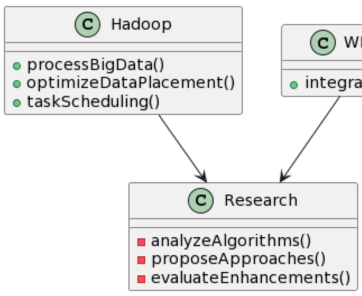

Fig. 3.2: Proposed Algorithm Processing Unit

efficiency is data transmission. Reduction in efficiency causes performance degradation of the system as shown in Figure 3.2. This methodology aims to improve efficiency of Hadoop MapReduce framework during parallel data processing and when operating in data-intensive environments. It intends to ameliorate response time and data throughput of system. It aims to maximize network load in Hadoop MapReduce for boosting the overall system performance. In this research work, initially data distribution is done, followed by speculative prefetching.

This research aims to optimize the network load in Hadoop MapReduce. It intends to improve the efficiency of Hadoop MapReduce framework during parallel data processing and when operating in data-intensive environments and to ameliorate mainly response time and data throughput of desired system.

Step 1: This step involves initialization of a series of entities such as data center, data locality and task scheduler.

Step 2: This step involves data distribution, prefetching and development of MapReduce. Data distribution is done according to node capacity and for prefetching operation, a speculative prefetching method is employed.

Step 3: This step involves Hadoop configuration and development of a locality-based scheduler and reduction of tasks into nodes using WELM-FF.

Step 4: This step involves performance inspection of proposed WELM-FF-Fair share approach and also comparison of proposed scheduling approach with benchmarking schemes.

A distributed Hadoop configuration with 20 nodes is employed and then job scheduling is accomplished fair scheduling scheme. Experiments are conducted on diversified Hadoop cluster with configurations. Further, heterogeneity within the cluster is quantified by considering several CPU or processor types, disk space and memory size. Hadoop 2.2.0 version is employed and configured with JDK 1.8 version and 128 MB block size. For input, a text file generated using java code is used. K-Nearest Neighbour (WELM-FF) based fair scheduler approach is employed for improving efficacy of Hadoop MapReduce model while parallel information processing and for enhancing system's response time and data throughput. In current methodology, performance of developed WELM-FF-based fair scheduling scheme is evaluated considering performance measures like job execution duration, throughput, performance rate and execution duration for clustering as shown in Figure 3.3.

In this proposed work, the developed WELM-FF scheduler's performance is assessed using afore-mentioned metrics and is compared with popular schedulers like capacity scheduler and default (FIFO) scheduler. Furthermore, performance of proposed WELM-FF scheduler is evaluated against different benchmarks like random text, sort, word count with regard to job execution time, performance rate, data locality, execution time for clustering and throughput through comparing WELM-FF scheduler as shown in Figure 3.4 and data locality-based FIFO scheduler. Also, superiority of proposed WELM-FF based fair scheduling scheme is compared with default scheduling method with regard to metrics like job execution duration, throughput, performance rate and execution duration for clustering.

3.1. Initialization. The entities like data center, data locality and task scheduler play a vital role in Hadoop MapReduce framework. The data locality indicates how close the input and compute data are (in common words, it indicates closeness of input data and computed data). Data locality contains distinct levels

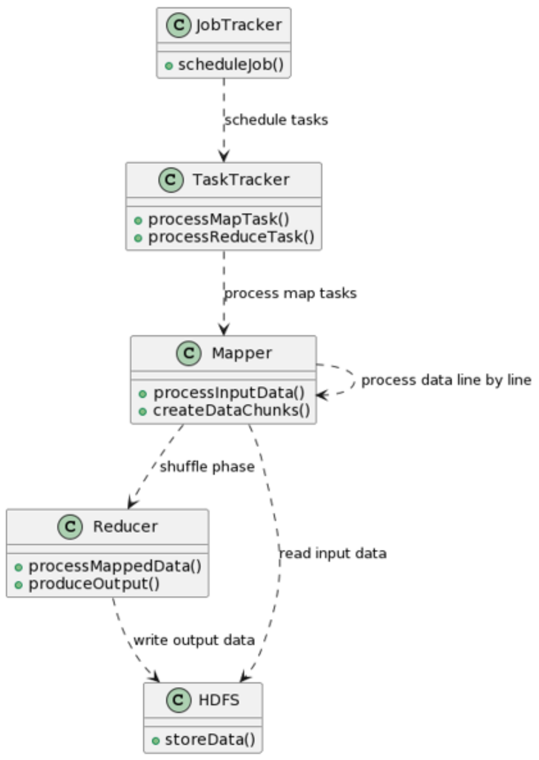

Fig. 3.3: Working of Map Reduce Technique

like rack level, node level, etc. In this methodology, data locality based on nodelevel is exploited. In data locality based on node-level, compute data and given/input data are coplaced on an alike/same node. In parallel data systems, data locality corresponds to an imperative element considered by schedulers. Data locality here indicates locality of input data. The task scheduler is associated with job scheduling in the Hadoop MapReduce framework. In proposed methodology, firstly these entities (data center, data locality and task scheduler) are intialized. The two text files generated using java code namely keyval and random text are used as input and applied to MapReduce job. Given an original matrix V (document term matrix), NMF seeks to approximate V as the product of two non-negative matrices W (basis matrix) and H (coefficient matrix), i.e.,

$$V \approx WH \tag{3.1}$$

This factorization can be expressed through the optimization problem:

$$\min_{W,H} \parallel V - WH \parallel^2 subjecttoW, H \geq 0 \tag{3.2}$$

where $\parallel \cdot \parallel^2$ denotes the Frobenius norm, ensuring non-negativity in the factorization. The resulting matrices W and H capture the underlying structures and relationships in the data. Reducing the dimension and converting the document word matrix into smaller ones are the main goals of this strategy, along with gathering alternate underlying structures within the data. Elements in these reduced matrices must be non-negative. Pattern recognition, text clustering, and bioinformatics are just a few of the several areas where the NMF method give the better results. Initially, input data or information is divided into several data blocks as per the node's processing potential within the cluster using the proposed scheduler as shown in Figure 3.5. Further, these information blocks are allotted to distinct nodes within cluster.

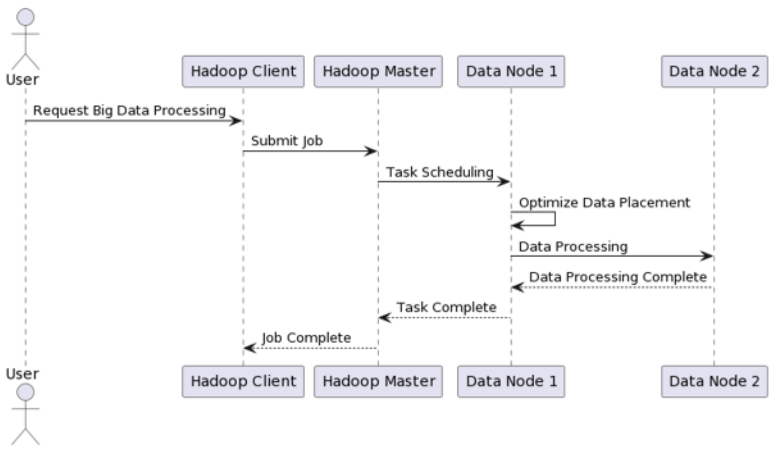

Fig. 3.4: Working of Data Localization

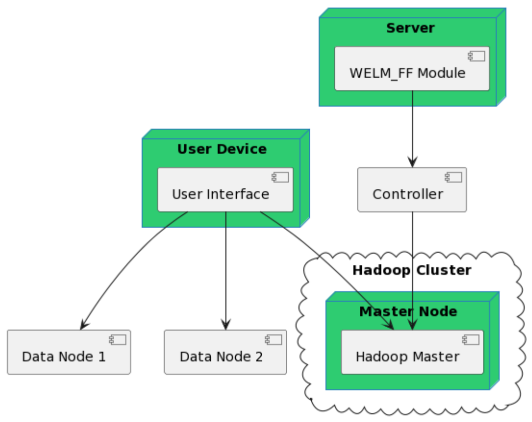

Fig. 3.5: Proposed User End Module

$$ODB \leftarrow DS/BS \tag{3.3}$$

where ODB indicates overall data blocks for input information, DS denotes input data's data size and BS denotes block size (default). Processing capacity (NPC) of each x node in the Hadoop cluster is estimated as,

$$NPC(x) \longleftarrow P(x) + MTE(x) \tag{3.4}$$

where NPC(x) indicates x^{2h} node's processing capacity, MTE (x) denotes mean task execution duration in x^{2k} node and P(x) indicates performance of $x^{\sqrt{k6}}$ node.

Within the cluster, NPC of all present nodes is determined through considering a) mean task execution duration of a specific job in specific node of cluster, b) combining CPU and existing memory of specific node at normal intervals. The mean task execution duration is determined through executing a specific job's few tasks on that node. Here, MTE is added to NPC as the job in distinct nodes is executed at distinct times and this aids in determining a node's NPC. Furthermore, percentage of CPU and memory utilization in each node of a diversified cluster can be determined. The free CPU and memory usage in each node within cluster is determined as

$$\begin{aligned}\text{FreeMem }(x) &\leftarrow ((100-\text{ UtilizedMem }(x)) \times Mem(x))/100\\ \text{FreeCPU }(x) &\leftarrow ((100-\text{ UtilizedCPU }(x)) \times CPU(x))/100\end{aligned} \tag{3.5}$$

The job tracker handles job scheduling. It allots tasks to idle task trackers in the cluster according to slot availability. Further task trackers process the reduce tasks and map tasks on a cluster's corresponding nodes. The mapper's job here is to typically process given/input data. This information (given data) is stored within the HDFS. The input file is further passed to the mapper function line by line. Further, mapper processes the data and creates several small chunks of data. Reducer's job here is to mainly process the data which approaches from mapper. After processing, it produces a new set of output, which will be stored in the HDFS. Between map and reduce phase shuffling and sorting is done. The information obtained from the map nodes is copied to the reducers node by shuffle phase. The sort phase sorts the intermediate information by key. where Mem(x) represents x^{sh} node's RAM capacity, UtilizedMem(x) represents percentage of utilized memory in x^{st} node of cluster, CPU(x) represents x^{st} node's processing capacity and UtilizedCPU(x) represents percentage of utilized CPU in x^{th} node of eluster. Further, performance of x^{th} node is determined through obtaining free existing memory and CPU from that node using equation (3.6) as

$$P(x) \leftarrow FreeMem(x) + FreeCPU(x) \tag{3.6}$$

Each time during job execution, the nodes are classified based on the NPC. The overall NPC of all nodes contained within Hadoop cluster is estimated using equation (3.7).

$$ONPC \leftarrow \sum_{x=1}^{n} NPC(x) \tag{3.7}$$

where n represents the number of data nodes and ONPC denotes the overall NPC. The amount of data blocks that are to be allotted for every data node i is estimated as,

$$NB(x) \leftarrow ODB \times (NPC(x)/ONPC) \tag{3.8}$$

where NB(x) denotes the number of data blocks allotted for x^{th} data node. For x^{th} data node, the number of data blocks will be allotted as per NPC using equation (3.9)

$$DN(x) \leftarrow NB(x) \tag{3.9}$$

where DN(x) represents the x^{2h} node in the Hadoop cluster.

Then the presence of map slots is checked on data nodes. Map tasks are allotted to nodes within the cluster if the slots are present (or available). In case the slots are unavailable, then the scheduler waits until the slots become available.

In Hadoop, typically the reduce stage does not begin until the map stage generates all intermediate information. It is necessary to wait until completion of all map operations in order to achieve the ultimate job output. Hence, after completion of execution by map task scheduler, the reduce task scheduler begins its execution. In reduce stage, the nodes are organized such that they possess reduce function as depicted in equation (3.10).

$$RN(x) \leftarrow RN_l, RN_2, RN_3, \ldots\ldots\ldots\ldots, RN_k \tag{3.10}$$

where RN(x) represents the reducer node or x^{th} node possessing reduce function, k denotes the number of nodes possessing the reduce function. Similar to map stage, the availability of reduce slots is checked in reduce stage. If reduce slots exist, then reduce tasks will be allotted to nodes. If slots are unavailable, then the scheduler will wait until the slots become available.

In order to identify outliers as shown in Figure 3.6, a boundary must be formed around them by merging closed contours. All the points within a single contour are grouped together under the same cluster by these contours, which are called cluster borders. Cluster boundaries of different sizes and shapes may be generated using this method, and it doesn't matter how the data is distributed.

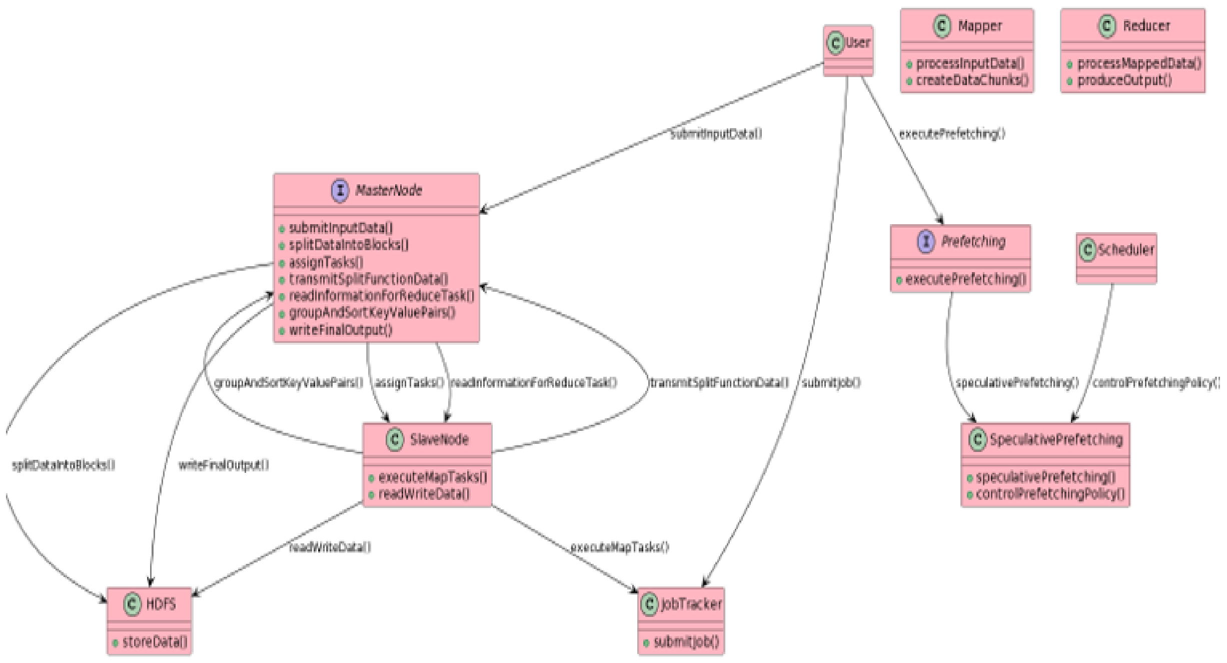

Fig. 3.6: Flowchart of Proposed work

The optimization problem associated with WELM can be expressed as:

$$\min_{R,a,\xi} R^2 + \frac{1}{\nu}\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N} a_i \tag{3.11}$$

subject to constraints:

$$\parallel \phi(x_i) - a \parallel^2 \leq R^2 + \xi_i,\ \ \xi_i \geq 0,\ \ \sum_{i=1}^{N} a_i = 1,\ \ a_i \geq 0 \tag{3.12}$$

where R is the radius of the hypersphere, a represents the center, and ξ_i are slack variables.

All of the points on a single contour fall inside the same cluster, which is why these lines are called cluster borders. By generating cluster boundaries of any shape and size independent of the data side, the suggested WELM technique surpasses the traditional approaches. The shortcomings of conventional methods may be remedied with the use of the WELM clustering strategy. There is currently no clear solution to the research topic of how to properly apply WELM during document clustering. There isn't a perfect document clustering strategy that fixes all the problems with the old ways and works better.

3.2. Hadoop based Data partition. The Map-Reduce architecture supports and underpins big databases with a lot of features, divides them into smaller sets, and then distributes them to different cloud groups for calculations. As seen in Figure 2, Map-Reduce views data as a key-value pair represented as <Key, Value>. At now, there is a lot of thought put into data-intensive processing, mostly centred on the Map-Reduce framework for investigating large amounts of data. Over time, it has evolved into a scalable and fault-tolerant data handling device that can calculate and process massive datasets with just a few of low-end computer cluster nodes. However, Map-Reduce does have certain inherent issues with its efficiency and implementation as shown in Figure 3.7.

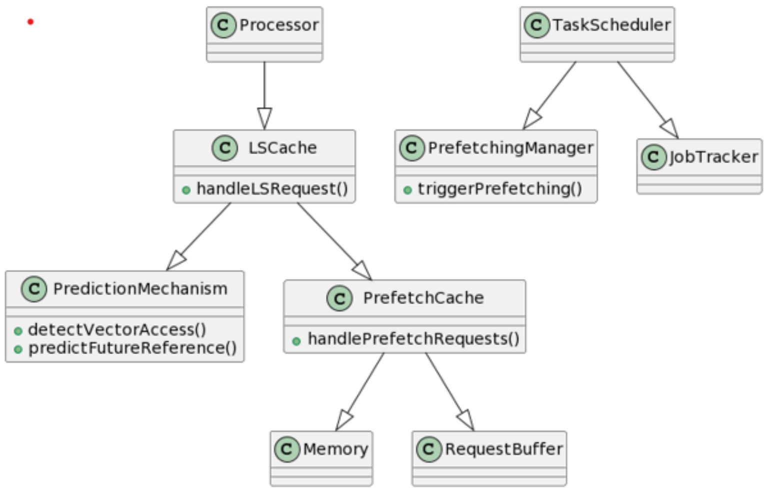

Fig. 3.7: Map-Reduce Framework with prediction mechanism

The fundamental goal of knowledge mining techniques is to extract valuable information from massive informational databases, often known as datasets. The information gathered from this procedure is quite useful for decision-making models. The computational competency and processing time are both negatively affected by the amount and dimensions of the data set as the intensity rises. Unfortunately, classical method parallelization is not straightforward, and many current optimization models aren't very effective at parallel calculations either.

One of the most important data mining categorization strategies is the decision tree. Experts in decision tree algorithms now focus on making data processing more efficient. Data sizes grow in direct correlation to the rate of advancement in system networking and real-time application development. The purpose of using a distributed decision-tree structure in tandem with the Hadoop framework to handle these massive data sets is to identify these concerns.

3.3. Biomedical XML document preprocessing. Due to the vast amount of medical document sets, the biomedical area is home to the most commonly utilised applications of text mining. For open access papers, the conventional text mining methods rely only on ”sorting and theme” for searching. Articles from open-access biomedical journals are licenced under the creative commons licence and are available in full text. In the first stage of the algorithm for extracting documents, it detects about 100GB of articles. For the extraction model to work properly, clear structured data is required. In most cases, all publications found in the PubMed repository include unstructured data in addition to XML tags. Articles' entire texts are pre-processed and clustered using the document clustering method. This method places a premium on abstractions, as opposed to previous biomedical text mining algorithms.

The clustering process, mathematically, involves grouping similar XML data, denoted by D_i, into clusters C_j. This can be expressed as:

$$C_j = \{D_i : similarity\,(D_i, C_j) \geq \text{ threshold } \} \tag{3.13}$$

where similarity measures the similarity between the XML data D_i and a cluster C_j, and a threshold is defined to determine when a document is considered part of a cluster.

When working with big and varied datasets, these clustering approaches are very important for improving the biomedical domain's information structure and accessibility. These days, XML is the standard format for online data exchange. Clustering has been the most popular and widely used method. In this context, clustering stands for the combination of XML data types and XML clustering applications that are comparable. Among them are query processing, web mining, document categorization, data integration, and retrieval of information.

The major issues in XML data preprocessing for clustering are described below:

- Initially, the clustering process calculates the similarity index among numbers of different XML data. Nevertheless, evaluating the similarity function is a major problem because of the heterogeneity property of XML documents.
- Implicit dimensionality has increased to a great extent.

Biomedical documents, phrases, sentences are used in the feature extraction to extract the main features of the original documents. The graph-based feature extraction generates the feature extraction by mining phrases or sentences from the set of key peer nodes of the overlay network. Finally, key phrases or sentences are extracted by computing the ranking scores and then selecting the highest scored phrases or sentences.

3.4. Traditional feature selection models on biomedical databases. In order to categorise the texts according to MeSH keywords, a new Bayesian Network (BN) method is created. Classification is carried out by this model based on the entity associations of various MeSH words, and it can readily depict the conditional independences between MeSH terms and the MeSH ontology. This method is used to depict the MEDLINE document resources at different levels of abstraction. In order to address these concerns, medical ontologies are used to discover the medical concept MeSH synonyms. Textual categorization schemes often use machine learning techniques in an inductive fashion.

Furthermore, a BN classifier based on support vector machines is used to categorise MEDLINE articles. When analysing the performance of a balancing approach, both BN and SVM-based BN classifiers become quite sensitive. Text classification and classification models are frequent areas where class-imbalance problems occur. The classic issue of class imbalance cannot be addressed by these methods. In order to make conventional categorization procedures more accurate, a lot of new, sophisticated methods have been created. Methods based on recognition, active learning, sampling, and cost sensitivity are all part of these generalised categorization systems. By excluding arbitrary data from the dominant class, sampling techniques have helped to address the issue of class imbalance. The term for these methods is under-sampling approaches. Oversampling occurs when there are insufficient new randomised data points for the minority class. For instance-specific document category classification, the cost-sensitive learning technique employs a cost-matrix. In order to learn rules, recognition-based approaches look at the minority class rather than the majority class's instances. To address the challenges posed by unlabeled data, active learning techniques are used.

In order to discover the hidden learning principles in the minority class examples, which could or might not employ instances from the majority class, feature selection based learning approaches are used. Automatic MEDLINE document classification is achieved by the use of Bayesian-based feature selection models, which provide a probabilistic framework. When a document has a particular MeSH label or is connected to the term's grandparents in the hierarchical model, Bayesian models do not work. For example, the words A01.047.025.600, A01.047.025.25, and A01.047.025.600 may all be used to represent the same document in an index that uses the terms A01.047.025.600.451. The MEDLINE documents can be better selected using an ontology-based document feature selection approach according to this assumption.

The suggested random sampling strategy involves selecting components at random and then removing them from the class that is overrepresented in the majority. This procedure is repeated until the minority class size and cost-sensitive learning method are equal. Among its components are adjustments to the relative cost of incorrect positive and negative class classifications. It is necessary to convert every page into a feature vector before using machine learning techniques for feature extraction. To address the problem that the high-dimensionality of the features space creates, many feature selection methods have been used. Feature selection for documents has long made use of several probabilistic and statistical machine learning algorithms. Neural network, K-nearest neighbour, Bayesian, decision tree, and symbolic rule learning are all part of it.

Scalable Machine Learning techniques and Rule-based reasoning methods provide the basis of feature learning models. Due to the lack of a prior classified example or restriction pertaining to data pieces having labels, this procedure falls under the umbrella of unsupervised learning. Ontology (Abox+Tbox) presents the categorization model. It learns on its own from massive datasets using scalable Machine Learning and Big Data methods. When it comes to categorization, feature extraction is a crucial step. We can classify all characteristics into two categories:

- Feature extraction is carried out on the small document sets by making use of noisy attributes and

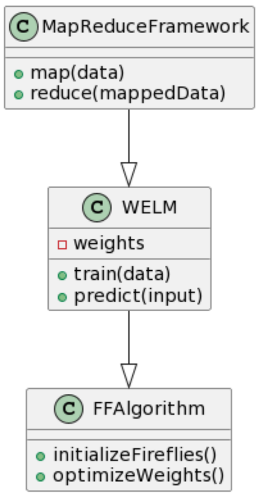

Fig. 3.8: Working of WELM-FF algorithm

contextual information.

In order to reduce the number of dimensions of high-dimensional features, traditional feature extraction algorithms do away with noisy features. Predicting medical diseases, such as cancer, genes, proteins, etc., has led to the development of several methods. When it comes to microarray cancer detection and prediction, the rule mining method is combined with PCA and back propagation. For improved performance, a modular neural network is designed to detect and assess heart illnesses utilising the Gravitational Search Algorithm (GSA) and fuzzy logic algorithm..

3.5. Weighted Extreme Learning Machine Technique (Weighted ELM) with the Firefly Algorithm. Incorporating weights into the learning process, the Weighted Extreme Learning Machine (Weighted ELM) method modifies the classic Extreme Learning Machine (ELM) algorithm. It creates a hybrid strategy for optimising the Weighted ELM shown in Figure 3.8 has parameters when coupled with the Firefly Technique, an optimization algorithm that draws inspiration from nature. A few formulae and notations depicting the Weighted ELM and its integration using the Firefly Algorithm are provided below.

1. Extreme Learning Machine (ELM): The traditional ELM formulates the output weights β as:

$$\beta = H^{\dagger} T \tag{3.14}$$

   Where:
   - H is the hidden layer output matrix.
   - T is the target matrix.
   - $H^{\dagger}$ is the Moore-Penrose pseudo-inverse of H.
2. Weighted Extreme Learning Machine (Weighted ELM): In Weighted ELM, the objective is to minimize a weighted objective function:

$$J(\beta) = \frac{1}{2} \sum_{i=1}^{N} w_i \left\| y_i - \sum_{j=1}^{L} \beta_j h_j \left(x_i\right) \right\|^2 \tag{3.15}$$

   Here, w_i represents the weights associated with individual training samples.
3. Combining Weighted ELM with Firefly Algorithm
   - Optimizing the weights $w - i$in the Weighted ELM is the goal of the Firefly Algorithm. Using the attraction between fireflies, the system adjusts the weights repeatedly. The following stages make up the method, which uses the objective function to assess attractiveness:

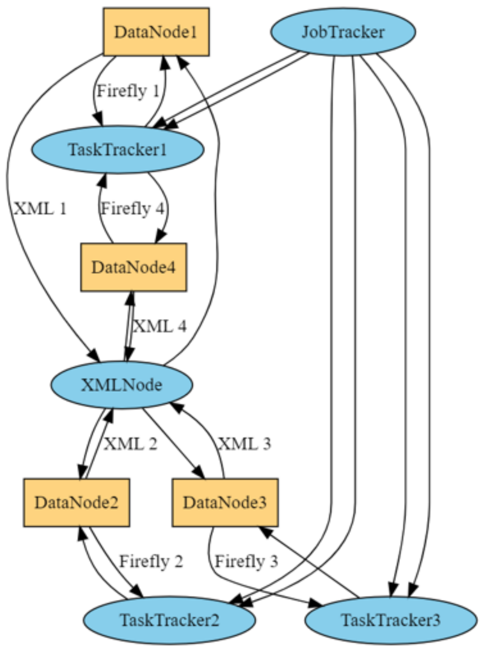

Fig. 3.9: Data Locality Based Firefly and WELM

- Start by setting the starting light levels and locations of the fireflies.
- Determine Attraction: Using the distance and light intensity of two firefly, determine the attractiveness between the two.
- Apply the attractiveness values to the weights and update them.
- Repeat: Continue iterating until you reach convergence or until you reach a predetermined limit. The precise formulation of the update equation for weights, $w - i$, depends on the particular specifics of the implementation and the issue environment, although it may be described using the Firefly Algorithm.

These methods' inefficient learning stages or coverage of local minimums lead them to be painfully slow. To further improve learning efficiency and performance, many iterative learning techniques are created. Based entirely on kernel-based methods, this strategy incorporates neurons in an ascending sequence. Unlike traditional Feed-Forward Network training, Bayesian ELM does not adjust the hidden layer. To reduce training error, all hidden layer parameters, including input and output weights and biases, are chosen at random. Slower network learning rates and less capacity to handle complexity are the main drawbacks of the aforementioned approaches.

From Figure 3.9, The two main steps of the WELM-FF (F-ELM) method are training and prediction. A three-tiered design is suggested for the training phase. In the training phase, scaling parameters in the hidden layer and weights in both the hidden and output layers are repeatedly generated, much like the WELM technique. Therefore, F-operations WELM's are comparable to those of the fuzzy inference system. To make predictions, a trained F-WELM algorithm takes an input feature vector and uses it to assess the outputs.

Using nodes and edges, decision trees may be shown as directed graphs. Always representing tests are the root and intermediate nodes, with outgoing edges representing node conditions. Additionally, various class labels, represented by leaf nodes, are accountable for uncovering latent patterns in massive datasets. The decision tree's decision patterns are predicted by these. The quantity of data lost during decision tree creation grows in direct proportion to the size of the k-trees. At each level, these models determine the decision criteria

Table 4.1: Amazon EC2 Setup Configurations

General purpose current generation	Linux/UNIX Usage (per Hour)	Windows Usage (per Hour)
m3.medium	0.0086	0.0591
m3.large	0.0162	0.1171
m3.xlarge	0.036	0.2201
m3.2xlarge	0.1539	0.4391
m1.small	0.0081	0.0261
m1.medium	0.0102	0.0531
m1.large	0.0175	0.1061
m1.xlarge	0.034	0.2111

by running an algorithm that is based on the hierarchical selection of characteristics. These are the main problems with this attribute selection method: 1) The amount of time and space needed for calculation also grows as the noise of characteristics in single and multi-leveled space increases. 2) Entropy measurements on the homogeneity of the information distribution are used by this model.

Figure 3.9 shows the results of using an WELM-based decision tree with a logistic regression function to identify infrequent decision rules; in this case, a new occurrence is arranged by climbing the tree from its root node to its leaf node, and a decision on the attribute is made based on the class attribute value of the leaf node through regression analysis.

If the database has a lot of dimensions, then can't use the WELM decision tree to build incremental trees. Also, using the Map-Reduce framework with mixed attribute types is not supported by this strategy.

Limitations

- Hadoop nodes are used and the memory limitations affect the tree's performance.
- Achieving greater performance on the part of the intermediate mappers requires fixing the static restrictions of the minimal class instances.
- Building the static classification and static parameters is a must during the training phase of a Map-Reduce system.

4. Performance Evaluation. For experimental evaluation, Hadoop with hardware configuration of 4 GB RAM, 250 GB hard disk and above 2 GHz processor is employed. Experiments are conducted on diversified Hadoop cluster with configurations as depicted in Table 1. The employed test-bed consists of one master node (job tracker and Name node) and twenty slave nodes (task tracker and data node). The heterogeneity within the cluster is quantified by considering several CPU or processor types, disk space and memory size on each node. Here, all nodes within cluster are connected using an ethernet switch. For execution, Linux operating system is employed. In proposed experiments, Hadoop 2.2.0 version is employed and configured with JDK 1.8 version and 128 MB block size. For input, a text file generated using java code is used. The master.txt file is used to declare master node name and workers.txt file to declare slave nodes' name. The file yarn-site.xml is employed to write node property and node values. Scheduler is controlled using xml files.

4.1. Implementation Setup . Amazon Elastic Compute Cloud (EC2) has three distinct storage tiers—small, medium, and large—each of which offers access to a unique set of cloud resources. You may access these instances and services across several time zones and countries. The pricelist for micro to big instance types for Windows Usage machines and LINUX/UNIX is shown in Table 4.1 for the Asia Instance servers.

The cloud's capacity to handle computations of varying sizes is made possible by Amazon Elastic Compute Cloud (EC2). For developers, EC2 means quick and simple web-scale compute. With no effort, you may configure the hardware capacity to your exact specifications using the accessible user-friendly interface. Instance addition and deletion are handled using the aforementioned interface. Customers may choose which EC2 instances to make public or private when they deploy these services to cloud storage servers. A pre-configured AMI, network permissions and security, and the uploading of a completed web application are prerequisites for implementing an Amazon Elastic Compute Cloud instance. Based on their computing demands, customers must choose the beginning and finishing points of the instance. The amount of AMI has to be present in several

Table 4.2: Memory Optimized - Current Generation

r3.large	0.0175	0.1741
r3.xlarge	0.0478	0.2821
r3.2xlarge	0.1785	0.4442
r3.4xlarge	0.48	0.6744
r3.8xlarge	0.8791	0.9583

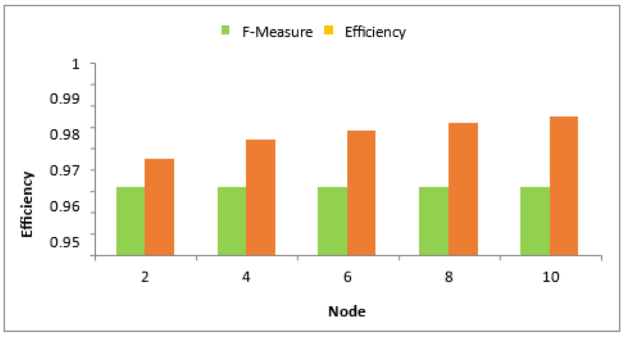

Fig. 4.1: Results of FF-RR's execution on 10 nodes in a Hadoop cluster

occurrences. The next phase involves configuring additional application services by monitoring the activity of newly formed instances. Both incoming and outgoing network traffic may be managed with the use of security groups and ACLs.

Various biomedical documents in XML format are subjected to experimental outcomes. The document types seen in the MEDLINE Repository vary among the various data sets. Table 4.2 is a visual representation of the results of the feature selection process's performance study utilising the Hadoop framework. In this case, we learn how long the Mapper and Reducer phases took to operate in the Hadoop environment by including the Rough-set approach and a Random Forest tree.

Visual representation of the feature selection process performance analysis utilising the Hadoop framework is shown in Figure 4.1. In this case, we learn how long the Mapper and Reducer phases took to operate in the Hadoop environment by including the Rough-set approach and a Random Forest tree. The chart clearly shows that, in comparison to other classification strategies that do not use a rough-set method, the Rough set with Random forest model's Mapper and Reducer stages have a less time-complex nature. Figure 4.1, clearly shows that in comparison to other classification strategies that do not use a rough-set method, the Rough set with Random forest model's Mapper and Reducer stages have a less time-complex nature. Analyses of the order models' performance are shown in Table 4.3 and Figure 4.2. When compared to other traditional Hadoop classification algorithms, Random Forest trees with roughsets have a far higher real positive order rate for biological datasets. The number of positive occurrences that represent assaults, as compared to negative ones, is shown by the true positive rate. The number of instances that were incorrectly categorised is shown by the error (percent). Also calculated are the outliers, expressed as a percentage, which show how many occurrences are unrelated to the present assault behaviour. Table 4.4 shows the classification models with True Positive and Error Rates metrics.

Figure 4.3 shows the performance metrics of Hadoop Models.In the first experiment, a regular Hadoop cluster (SHC) showed a 120-minute processing time, 500 records/minute throughput, and 95% accuracy. Reducing execution time by 80 minutes, increasing throughput by 750 records per minute, and improving accuracy by 98% were all outcomes of Experiment 2, which included adding further nodes to the cluster (ECWAN). Even though the execution time went up to 100 minutes in Experiment 3, which focused on Hadoop with Improved Data Compression (HWIDC) data compression approaches, the throughput remained competitive at 600 records

Table 4.3: Classification models for feature selection using Rough-set

Algorithm	Mapper Time (mins)	Reducer Time (mins)
GA_FSA	15.35	4.57
Neural Network	21.53	5.75
WELM-Tree	14.65	5.36
WELM with firefly	12.43	4.567

Table 4.4: Classification models with True Positive and Error Rates metrics

AModel	True Positive (%)	Error (%)	Outlier (%)
GA_FSA	81.45	25.67	12.54
Neural Network	83.157	21.56	15.37
WELM-Tree	84.26	27.89	16.24
WELM with firefly	89.45	19.4	11.15

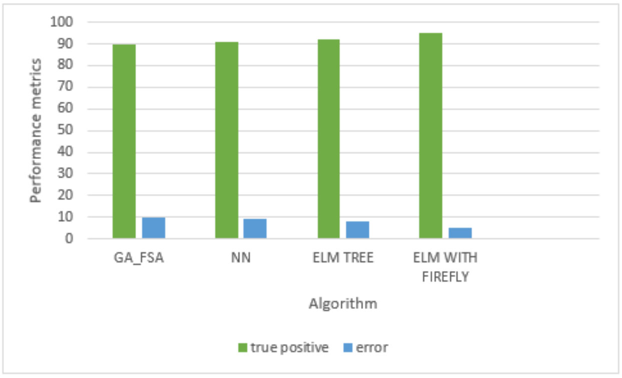

Fig. 4.2: Performance metrics for classification models

per minute and the accuracy rate stayed at 96%. These findings demonstrate how important it is to optimise Hadoop-based systems using data processing methods and cluster design. Improving performance was a direct result of the tweaked cluster design, and investigating data compression methods revealed promising new ways to strike a balance between the execution time and accuracy trade-offs. Improved distributed data processing using the Hadoop architecture may be possible with more research and development.

5. Conclusion. The proposed work differs from existing solutions in that it does not require learning the classification accuracy for certain types of information attributes earlier, which is not always possible in practise. Instead, it takes an exponentially decreasing size of distributed databases, data pre-processing, and the classification true positive rate. The distribution noise or anomaly problem, mining sparse, scaling up for high-dimensional space, and static optimization are the main challenges with standard data mining algorithms when applied to massive data. This chapter details the application of a new decision tree model based on the toughest criteria to biomedical datasets using the Hadoop framework, with the goal of improving the computation error and true positive rate. Lastly, a potential strategy for improving ML models is the proposed Weighted Extreme Learning Machine (Weighted ELM) in conjunction with the Firefly Algorithm. To make the classic Extreme Learning Machine even more sophisticated and adaptable, the Weighted ELM adds weights to it. The compatibility of the hybrid model with the Firefly Algorithm, a method of optimization that draws inspiration from natural behaviours, allows for more efficient parameter tuning. The capacity to account for the significance of individual training samples by using weights is the main benefit of the Weighted

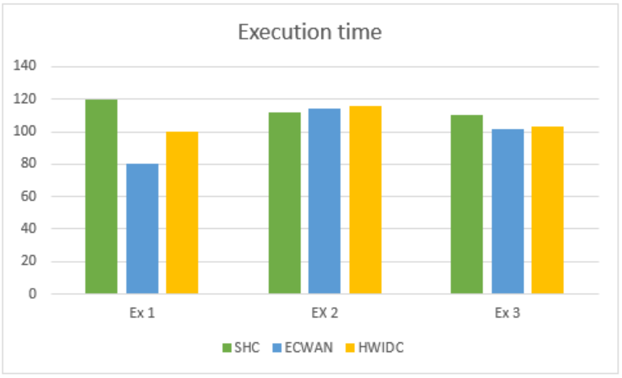

Fig. 4.3: Performance metrics of Hadoop Models

ELM. This makes the model more flexible and resistant to data distribution issues, which is especially helpful when certain samples are more important than others for learning. By mimicking the characteristics that make fireflies appealing, the Firefly Algorithm improves the Weighted ELM's optimization. Improving the model's convergence and overall performance, this optimization technique helps locate ideal weight combinations. Nevertheless, there are a number of variables that affect how well this hybrid strategy works, such as the parameters used, issue specifics, and dataset characteristics. To get the best outcomes for particular applications, it is vital to conduct thorough experiments and fine-tune. Addressing complicated issues where flexibility, efficiency, and resilience are vital, the Weighted ELM with Firefly Algorithm shows promise in practical terms. The method's competitiveness across a larger variety of applications may be determined by future work that investigates hyperparameter tweaking, experiments with varied datasets, and compares with other state-of-the-art approaches. When combined, Weighted ELM and the Firefly Algorithm provide new possibilities for improving machine learning models and expanding their use in many fields.

6. Acknowledgement. This study was financially supported by Chonnam National University (Grant number: 2023-0926).

REFERENCES

[1] Zhao et al, *A data locality optimization algorithm for large-scale data processing in Hadoop*, In 2012 IEEE Symposium on Computers and Communications (ISCC), pp. 000655-000661. IEEE, 2012.
[2] Zhang et al, *Improving data locality of mapreduce by scheduling in homogeneous computing environments*, In 2011 IEEE Ninth International Symposium on Parallel and Distributed Processing with Applications, pp.120-126, IEEE, 2011.
[3] Zhang et al, *An improved task scheduling algorithm based on cache locality and data locality in Hadoop*, In 2016 17th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT), pp.244-249, IEEE, 2016.
[4] Zhang et al, *A comprehensive bibliometric analysis of Apache Hadoop from 2008 to 2020*, International Journal of Intelligent Computing and Cybernetics, vol.16(1), pp.99-120, 2023.
[5] Zhai et al, *The Emerging' Big Dimensionality*, IEEE Computational Intelligence Magazine, vol.9(3), pp.14-26, 2014.
[6] Zaki et al, *Parallel algorithms for discovery of association rules*, Data mining and knowledge discovery, vol.1(4), pp.343-373,1997.
[7] Zaki et al, *CHARM: An efficient algorithm for closed itemset mining*, In Proceedings of the 2002 SIAM international conference on data mining, pp.457-473, 2002.
[8] Yuan et al, *Decision tree algorithm optimization research based on MapReduce*, Software Engineering and Service Science, 6th IEEE Int Conf in 2015 1010-1013, 2015.
[9] Yu et al, *Towards scalable and accurate online feature selection for big data*, In International Conference on Data Mining ICDM, pp.660-669, 2014.
[10] Yousef et al *Combining multi-species genomic data for microRNA identification using a Naive Bayes classifier*, Bioinformatics, vol.22(11), pp.1325-1334, 2006.
[11] Yongjiao et al, *An OS-ELM based distributed ensemble classification framework in P2P networks*, Neurocomputing, vol.74(1), pp.2438-2443, 2011.

[12] Yoan et al, *SOM-ELM—Self-Organized Clustering using ELM*, Neurocomputing, vol.165, pp.238-254, 2015.
[13] Yilmaz et al, *A new modification approach on bat algorithm for solving optimization problems*, Applied Soft Computing, vol.28, pp.259-275, 2015.
[14] Xia et al, *Blaze: A High-performance Big Data Computing System for High Energy Physics*, In Journal of Physics: Conference Series, Vol. 2438(1), p. 012012, 2023.
[15] Sharma et al, *New efficient Hadoop scheduler: Generalized particle swarm optimization and simulated annealing-dominant resource fairness*, Concurrency and Computation: Practice and Experience, vol.35(4), e7528, 2023.
[16] Sharma et al, *A review on data locality in hadoop MapReduce*, In 2018 Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC), pp.723-728, 2018.
[17] Oliveira et al, *Dynamic Management of Distributed Machine Learning Projects*, In Intelligent Distributed Computing XV, pp. 23-32, 2023.

Edited by: Polinpapilinho Katina
Special issue on: Scalable Dew Computing for future generation IoT systems
Received: Jan 31, 2024
Accepted: May 15, 2024