Hadoop Data Locality Performance Enhancement in Heterogeneous Computing Environment

Dr. Vasantha M[1*], Dr. GirijaV[2]
[1]*Associate Professor*
Department of CSE (IoT & Cyber Security including Blockchain Technology)
Cambridge Institute of Technology, An Autonomous Institution Affiliated to VTU
K. R. Puram,Bangalore-560036, India

[2]*Associate Professor*
Department of CSE
Cambridge Institute of Technology, An Autonomous Institution Affiliated to VTU

Abstract

It is required for the scheduling of Map-Reduce activities on Hadoop clusters to have data localisation, which is provided by the Data Placement (DP) mechanism of the Hadoop Distributed File System (HDFS). The Hadoop job scheduler is an independent piece of software that can be tailored to the requirements of every particular company via the use of various alterations. The three schedulers that are currently available for Hadoop are referred to as first-in, first-out (FIFO), compute capacity scheduling, and fair scheduling policy. Each of these schedulers uses a strategy for task allocation that takes into consideration the data's geographical proximity. They are not applicable in all situations and do not provide sufficient assistance with data localization when making plans for scheduling. On the basis of these discoveries, we suggested a method for task scheduling that makes use of data localisation and the concept of resources-prefetching. We suggest developing a new approach of clustering that we will refer to as Map-Reduce-based Firefly with Ridge Regression Based Scheduling (FF-RR). This will allow us to more effectively arrange enormous volumes of data. We are able to preselect possible nodes for task allocation by comparing the amount of time that is still needed to finish a job to the amount of time that is required to send a resources block. After that, we choose a non-local map task from the unfinished work queue to perform as a prefetching operation. Find out which home node in the network is suggested for a certain map job, and then have the necessary resource blocks sent there from the network node that was determined to be the nearest to those resource blocks. For our purposes, ensuring the location of the data in this manner would be sufficient. In conclusion, we carried out an experiment to demonstrate how the technique of prefetching resources helps to enhance the localization of work data and reduces the amount of time necessary to do a job. The FF approach increases both the quality and performance of clusters when it is employed inside an environment that supports Hadoop. The findings of the tests demonstrate that it is more effective than other contemporary methods.

Keywords: Hadoop scheduling, data locality, resources-prefetch, Ridge Regression, Firefly.

1. Introduction

In May of 2012, researchers were concurrently making progress in a broad number of other areas while also ensuring that Hadoop 1 remained operational. Every few months, a new release version 2.7.2 is made available for download. During the process of upgrading to the next major version, significant parts of Hadoop were altered in order to include newly developed capabilities. The differences in the design of Hadoop between versions 1 and 2 are shown in Figure 1. One of the most important differences is that each Map-Reduce task is implemented as its own independent piece of code. Yet Another Resource Negotiator, or YARN for short, is the moniker given to the system that is in charge of distributing and managing these resources. The structure of the HDFS file system also underwent some minor adjustments. These modifications have made it possible to fix the problems that plagued Hadoop version 1.0. [1]. The programming paradigm known as Map-Reduce is one of the cluster nodes that makes it possible to scale to a high degree. It makes the fundamental assumption that the whole ideology is consistent and operates on the same plane from the very beginning [2]. Despite the fact that Apache Hadoop is only one of many different possible implementations of Map-Reduce, Map-Reduce is nevertheless an extremely important component of the framework.

Map-Reduce is able to assist stakeholders in improving the system's dependability and performance [3] by performing parallel analysis of large amounts of raw data distributed over a huge number of clusters. The cluster is often responsible for tasks of the map and decrease variety. The high-level structure of the Map-Reduce algorithm is shown in figure 1.8, and stages 1 and 2 of the method are explained in reference 4. Phases: Steps in the Mapping Process: • Reducing the size of the input data comes from HDFS in chunks at a time. Utilizing the map task line allows for the processing of each block in isolation. The map function that the user has picked is then responsible for constructing the output data as key-value pairs [5]. The data is buffered, organized, and recorded to disc in a sequence of spill files during this step. These spill files will subsequently be integrated into the final map output file.

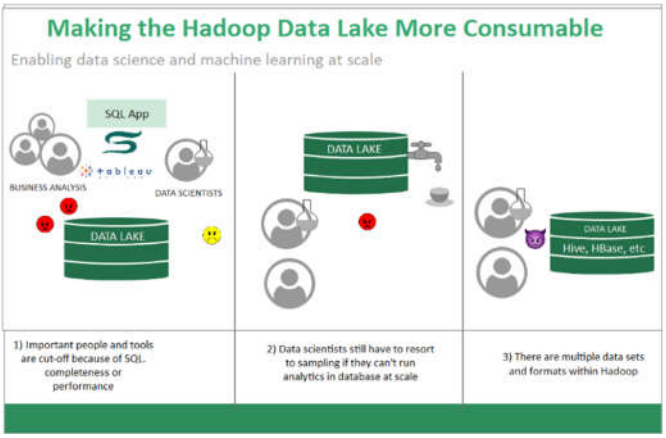

Fig.1 Hadoop Architecture

1.1 Background

In today's world, there is an ever-increasing quantity of information that can be discovered by searching the internet. The terms "unstructured data" and "no present format" refer to the same thing; examples of "unstructured data" include documents, books, photos, emails, diaries, and videos; examples of "semi-structured data" include character arrays, integers, and characters. The vast majority of data are unstructured, which renders standard ways to maintaining databases ineffective.

By merging several datasets (geolocation, weather, and traffic), vast amounts of data that are now being underutilized in a number of different domains might be put to greater use. According to the Heterogeneous, Autonomous, Complex, and Evolving (HACE) theorem [6, the defining characteristic of Big Data is as follows: [Citation needed]

A few examples of heterogeneous data sources are instant messaging services, social networking websites, professional networks, and microblogging platforms like Twitter and Facebook. The ability of Big Data to drive itself autonomously is a game-changer.

Controls in this system are decentralized and dispersed, which ensures that individual data sources are able to function independently of one another. Every server that makes up the World Wide Web (WWW) [7] is capable of creating data and operating its services without the intervention of any other server. Big Data is notoriously difficult to manage as a result of its complexity, and it has the potential to rapidly destabilize management systems that are dependent on it. Autonomous servers offer quick responses and limitless service for a select group of Big Data applications, such as social networks and search engines like Google. • Large amounts of parallel processing or distributed data, as well as a wide variety of data sources (sequential, multi-source, multi-table, multi-view, etc.), add to the complexity. • Autonomous servers offer fast responses and limitless service for a select group of Big Data applications, such as social networks and search engines like Google. When the complexity of the data and the real treatment techniques increase in tandem with the amount of the data, it becomes impossible to satisfy demands such as analysis, storage, and capture by using relational database systems. Typically, complex data will display some type of dynamic shift throughout the course of time. The study of big data is now through a period of fast growth. For example, a remark on a social media platform has to be scraped within an hour of the moment it is published for the first time. As a result, the procedure has to be effective, and all pertinent information needs to be included [8]. In order to keep up with the ever-increasing demand for data, the effectiveness and usefulness of the strategies and technologies that are now in use need to be improved. We will need to adopt a new approach if we wish to improve our capacity to analyses and make use of Big Data without allocating more resources to the endeavor. The exponential increase of data has left traditional techniques of data mining incapable of keeping up with the demand for the service. A processing model has to take into consideration the inherent heterogeneity, unpredictability, and complexity of the data in order for it to be able to effectively manage enormous amounts of data.

1.2 Motivation

[9] The HDFS metadata is stored on a separate computer that is referred to as a Name Node. The information pertaining to the program is stored on several servers that are referred to as "Data Nodes." Each server establishes a connection to every other server and shares data via TCP protocols. The data durability of HDFS is not guaranteed by the usage of data protection methods like those found in Luster and PVFS by the Data Nodes in HDFS. In the same way as GFS, many Data Nodes store identical copies of all data. When a file is stored in HDFS [10], it is first broken down into several smaller, identical sections known as "data blocks," which are then spread around the cluster. We make a copy of each data block to safeguard against the possibility

of any information being lost in the event that a node-wide or rack-wide connection fails. In addition to this, Hadoop functions as a proxy for the HDFS block replication mechanism, which is aware of racks. Because of the replica placement strategy, the first of three copies of a data block is constantly kept on the node that is responsible for hosting the writer.

The maximum number of copies that are ever stored in a single rack is always two, and the remaining copies are randomly distributed among an extra node or two. In contrast to conventional file systems, HDFS provides access to a programming interface (API) that reveals the locations of block storage. One application that might potentially benefit from this is the Map-Reduce framework for job scheduling. This could potentially increase retrieval performance. In this manner, a modification to the replication factor of the file may

Likewise be made by a program. The replication factor for HDFS is set to three by default. A replication factor of more than three is advised for use with files that are often read or have a size that is much larger than average. To summarizes, the physical placement of the blocks and the copies of those blocks is very important for guaranteeing consistent performance of both HDFS and Map-Reduce [11]. The following is how the outline of the paper looks: In Section 2, we take a look back at research that has been done before. In Section 3, we detail the research approach that was used for this study. In Section 4, we present and discuss the results of the experiments, as well as their analysis, and lastly, in Section 5, we provide some suggestions for research that may be done in the future.

2.Related Work

DP's utility for Map-Reduce workloads on the Hadoop cluster has been the subject of investigation in a number of studies. The DP technique is shown across the following publications. Throughout staged predictive inspection, the authors of [12] proposed a solution to the issue of duplicate data files that occurred throughout the process. The predicted consumption of any relevant document may be combined with the probability hypothesis to construct an approach that is comparable to the replication strategy. If the necessary papers are easily accessible, you may use this approach to replicate the documents that are utilized the most. An erasure code has been added to the absolute least amount of information that can be recovered. Therefore, in comparison to the standard way, this method improves convenience while simultaneously increasing levels of safety.

The goal of the new technology known as Data-Grouping-Aware (DRAW), which was created by [13], is to improve the efficiency of data-intensive applications in regions where there is concentrated user attention. DRAW investigates the idea of recovering information from framework log data in great detail. It breaks apart the information and then reassembles it into ideal clusters in such a way that the most parallelism possible for each group may be achieved without sacrificing load balancing. Using a test bench with forty nodes and a wide variety of DP approaches, we get to the conclusion that DRAW increases the throughput of local map operations for two Map-Reduce applications that have been verified. When compared to Hadoop's default random placement, it increases overall execution speed and reduces the amount of time required for the map step. We provide an overview of the Snakelike Data Placement technique, sometimes known as SLDP [14], which may be used across all of Hadoop's nodes. To get started, SLDP chooses a method that can divide many nodes into numerous VSTs while still taking into consideration the heterogeneity of the network. After then, each VST uses a mechanism that involves going around in circles, taking into consideration the data heat, in order to disperse the information blocks across the nodes. SLDP not only makes use of hotness-associated replication to reduce the amount of data storage space required, but it also offers an effective capability for managing power. Experiments conducted with two datasets taken from the real world shown that SLDP has the ability to improve Map-Reduce performance while simultaneously using less space and energy [15].

We provide a Secure HDFS (Sahds) [16] that modifies the SFAS section designation process that is used by the HDFS in Hadoop. This helps us address the problem of duplicating data without the risk of losing any of it. When the Sahds plan is applied to a file that has been uploaded to our Hadoop system, it generates a neutral summary of prospective nodes in groups. This summary is then sent to the new Hadoop DP policy.

The contents of the file will be distributed over several nodes of the same type via Sec HDFS, and the replications of the file will be preserved whenever it is possible to do so. This study, in contrast to other previous efforts on DP, does not measure the amount of time required to complete the stated task or the amount of time required to complete the work without failing. This will come in handy in the case that Hadoop's execution is interrupted by a problem at some point. The DP approaches that are now in use lead to an increase in overall performance that is 36.4% better. By adopting a flexible DP method, we were able to improve upon previously used methodologies.

[17] HDFS has unveiled its hybrid mechanism, which combines the use of traditional hard discs with solid-state discs in order to improve performance while minimizing the amount of additional energy that is required. In comparison to hybrid setups, the suggested technique demonstrated an enhanced decrease in energy consumption of up to 20% when bigger datasets were used. We offer a standardized information insertion approach that takes security concerns into consideration, and it is intended for use in scientific activities that may be carried out in the cloud. In this strategy, the safety of the data is ensured in three different ways. The confidentiality of user information, the accuracy of user data, and authenticated access are some examples. Calculating the cost of the security services for a data center may be done with the use of a security model. After that, an algorithm called Ant Colony Optimization (ACO), which is a dynamic approach, is used to choose the most effective data

centers. [18] This paper presents a unique concept in which vital data is kept in a cloud that is both trusted and private, while less sensitive material is kept in a cloud that is both public and less safe. This arrangement enables data to be retrieved quickly while still retaining anonymity. Sadly, the frameworks that were already in place for hybrid clouds did not make it feasible to extract data while also maintaining the identity of users. As shown in [19], Last-HDFS is an example of a distributed file system that has a difficulty with the placement of data. This permits location-aware cloud storage and the loading of files according to rules. In addition, it guarantees that the location policy is adhered to, even if policy compliance is disturbed by data replication or load balancing. This is achieved by ensuring that the location policy is followed to.

The process of organizing data known as clustering involves locating and then sorting out groupings of things that have similar characteristics. When considered individually, such individuals share the fewest features with one another. The primary objective of clustering is to determine with high precision, for each item in the data, which cluster it should be placed in. Concerning the problems caused by clustering, some potential solutions have been proposed [20]. The K-means strategy is popular in the scientific world due to the fact that it is easy to understand and use. The location of the initial centroid in the issue space is very important to the success of the k-means clustering solution. Because this strategy often converges with local optima near to the starting point of the search, the process of arriving at the global optimum solution may need a greater number of iterations. In recent years, various bio-inspired algorithms have been created in order to solve optimization difficulties that have been present in a variety of scientific and technical domains. The Genetic Algorithm (GA), Differential Evolution (DE), and Particle Swarm Optimization (PSO) are a few examples of approaches that are used to handle clustering problems that rely on evolutionary computing. Predictions of the implementation of a GA-based method referred to as "GA clustering" are made in [21]. Within the feature space, it is feasible to discover significant cluster centroids by using GA's search capability techniques. The anticipated DE at the [22] level influences the clustering approach that will be used. To get over the challenges of clustering, the ACO approach [23] makes use of dispersed agents. This is analogous to the way that actual ants locate food when it is far from their nest. Using a PSO-based algorithm, [24] constructed a model that took into account elements that are common to both bird flocking and other forms of fish schooling. You are free to use this strategy with or without having prior knowledge of the necessary cluster size. Even in this day and age, it is challenging to cluster huge datasets that include files of a very high size. A number of the sequential clustering strategies have a significant computational cost, both in terms of the amount of storage space required and the amount of time it takes to carry out the clustering. In light of these concerns, it is very necessary to make use of a technique of clustering that makes the most of parallel and distributed computing in order to have access to the massive data sets.

In recent years, the Map-Reduce programming model has become an increasingly popular alternative to other, more traditional techniques of parallel data processing programming. It is a creative solution to the problem of large-scale clustering since it automatically parallelizes the assignments and provides a load balancing and fault tolerant paradigm. This makes it an original approach to the problem. In a distributed system, several different approaches based on the Map-Reduce paradigm have been projected as being useful for the clustering of data. [25] Provides evidence that proves the scalability of these difficulties when used on a massive scale by using GA to cluster the data inside the Map-Reduce architecture. [26] Utilizing the k-means parallelization technique inside a Map-Reduce context allowed for the development of a clustering algorithm that performs admirably with enormous datasets. K-means and PSO are both algorithms that are reliant on Map-Reduce. Parallel K-PSO is a unique approach that was presented in [27] to improve clustering results and increase processing capabilities for enormous amounts of data by merging the K-means and PSO algorithms.

Scalable MR-CPSO was developed by [28] using the parallel Map-Reduce methodology in order to address the issues that arise with the PSO algorithm when it is used to extremely large datasets. In the article [29], it is said that the k-NN approach, which is based on the Map-Reduce paradigm, will be useful in a distributed computing scenario for the processing of huge amounts of data. As part of this investigation, we provide FF-RR, a Firefly system that is based on Map-Reduce and is designed to cluster enormous datasets in an effective manner. The Map-Reduce programming model is used to the construction of Firefly inside a Hadoop environment. During the process of organizing massive data samples into cluster groups 56, the primary objective is to reduce as much as possible the total Euclidean distance [30] that separates each data instance from the cluster centroid. As a consequence of this, the FF-RR approach makes the clustering of massive data sets easier without compromising the quality of the final output. In addition to that, a comprehensive investigation of the various approaches to data localization is provided. In addition to this, we go through the ways in which HDFS maintains the safety of data and cuts down on energy use.

3. Proposed Modelling

The FF-RR data clustering tool is based on Firefly and was developed using the Map-Reduce programming language. During the process of grouping together several examples of data into clusters, the primary objective is to reduce as much as possible the total Euclidean distance that separates each instance of data from the cluster centroid. As a consequence of this, the FF-RR approach makes the clustering of massive data sets easier without compromising the quality of the final output.

In response to the exponential development in the quantity of data that is being collected all over the world, a computer technique known as "data mining" was created as a way to analyse, discover new patterns, and draw better conclusions from the massive volumes of information that are collected. In the field of information technology, this need is rapidly becoming more

prevalent and important. This is due to the fact that producing a major evaluation in a short amount of time is very difficult, if not completely impossible. Effective data mining strategies are required before automating the data analysis process.

The process of dividing a set into smaller subsets (also known as "clusters") whose members have the most characteristics in common is referred to as "clustering." When broken down into its component components, it displays the fewest possible points of agreement that can be reached between those parts. The primary objective of clustering is to determine with high precision, for each item in the data, which cluster it should be placed in. In the past, there were many different strategies put up to cope with clustering concerns; however, none of them were successful. Because it is so straightforward to put into practise, K-means has found widespread use in the world of science.

The outcomes of K-means clustering are highly dependent on the starting centre that is chosen for the cluster. Because this technique often converges to local optima close to the starting point of the search, it could need further iterations before arriving at a solution that is globally optimum.

3.1 Improved Firefly Algorithm

The following straightforward and idealised criteria are adhered to by FF: • two fireflies of different sexes may be attracted to one another due to the gender differences between them; • the firefly's attraction and brightness diminish with distance. Therefore, the firefly with the more bright light may be drawn to the one with the less brilliant light, and vice versa. • Fireflies keep their brilliant light regardless of where they are or what they are doing. This behaviour occurs only when there is no other bright choice available to the firefly.

The fluctuation in brightness that occurs with increasing distance is shown by the symbol

$$\beta = \beta_0 e^{-\gamma r^x} \qquad (1)$$

Since the attractiveness of an FF is related to the quantity of light experienced by neighboring FF, we may express this as where _0 is the attractiveness at r=0. The mobility of an FF i attracts the attention of more attractive (brighter) FFjis, and vice versa.

$$x_i^{t+1} = x_i^t + \beta_0 e^{-\gamma r_{i,}^2}\left(x_j^c - x_i^t\right) + \alpha_t \epsilon_i^t \qquad (2)$$

Whereas x_j^c is a uniformly distributed vector of random counts derived from a Gaussian distribution at time t, the second term reflects the attraction. It is just a random stroll if $\beta_0 = 0$.

3.2 Data Clustering in Firefly Algorithm

Within the framework of the FF-RR methodology, the cluster centroid is altered in two steps before the fitness of the population as a whole is evaluated. The objective is to locate the optimal answer to the problem of minimising the Euclidean distance squared that separates each instance from the core of the cluster. Instead, we seek the optimal grouping of components or data items by minimising the total squared Euclidean distance between them, using equation (1) as a fitness function in FF to do so. Acquiring a Map-Reduce-based FF for data clustering is shown in Figure 2, which shows the acquisition.

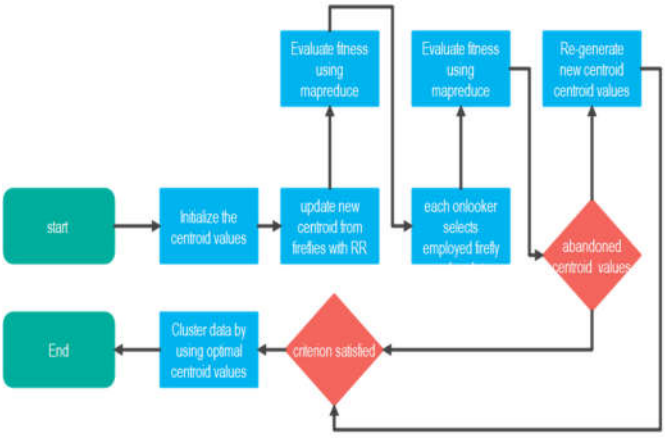

Figure 2: Map-Reduce-based FF

Figure 2 depicts the process that should be followed when finding the values of the centroid. These guiding ideas are the answer to the problem with the associated FF. One metric that may be used as a rough estimate of cluster quality and solution fitness is the sum of the squared Euclidean distances. This can be done by multiplying each distance along the Euclidean axis. The superior FF will inform the inferior FF of any new developments and will reveal its one-of-a-kind solution to the problem. A more astute FF will choose the solutions that result in improved fitness, include the necessary alterations into the solution, and then revaluate the fitness of its own solutions. It is a recursive process, and it terminates when the number of iterations hits the limit that was previously specified. If, after a certain length of time, an existing solution does not bring about an increase in the fitness range, the FF will develop a brand new solution. Working with a large dataset, on the other hand, will lengthen the amount of time needed to calculate the fitness value. Map-Reduce is used in order to generate an approximation of the fitness value.

Algorithm 1 Function of Firefly Optimization

RecordID Key Value Function Map
For each firefly in Fireflies,

we first read its ID.

a record in Fireflies.
end for

firefly_ID	Extract_FireflyID\\\(firefly\\\)
CV	Extract_Centroids\\\(firefly\\\)
min_Dist	ReturnMinDistance\\\(record, CV\\
centroid_ID	i
new_Key	(firefly_ID, centroid contains minDis
new_Value	(mm_Dist\\\)

end.

final function

Reduce (minDist,1) is in the collection of values for the key.

(Firefly_ID, centroid_ID).

Initialization:

Average = 0 count = 0 sum_Dist = 0

For all values in value_list, set dist to 0.

Sum_Dist = sum_Dist + min_Dist

end for

average = Dist = ExtractMinDist(value)

count = count +1

Distance Equals the Summing Up of Dist/count.

Function Termination: Emit(key, average Dist)

As we have seen, the HDFS map function saves the data that is required in order to determine the location of the cluster centre while performing the FF operation. This function will extract the centroid values for each FF, compute the distance along the centroid ID that separates each data record from the centroid rates, and then return the value that corresponds to the distance that is the shortest. The map function creates a new composite key and value by using the FF_ID and the centroid ID as identifiers for the minimal distance between two points. As a direct result of this, a brand-new key and discount percentage are produced for each transaction. After then, it arranges all of the distances and rates according to the same key. The key is built as a fitness range for each FF by making use of the reduction function's given shared distance in its construction.

In its most basic form, regression may be described as a set of methods that are used to improve the statistical fit of functions. The classification of these methods is heavily influenced by the function that was used in order to fit the data. In the context of linear regression, the linear function that is commonly employed to describe the link between the vector independent variable x and the scalar dependent variable y is referred to as the regression line. The following is an example of one form that may be used to represent such a function.

$$f(x_v\delta) = \delta_0 + \delta_1 x_1 +, \dots, +\delta_M x_M \quad (3)$$

$$f(x,\delta) = s^{\tau}x \quad (4)$$

Since it is assumed that f is linear, the parameters _j,...,_N are referred to as the slope parameters, and _j is the origin-offset parameter.

$$y = f(x,b) + er \quad (5)$$

When 'er', the error-term, is a normally distributed random variable with zero mean and unknown variance σ'.

$$E(y_1) \quad f(x,\delta) + E(er) \quad (6)$$

the expected value, fi, is where. Least Squares (L.Ss) is a popular technique for estimating the parameters () of linear regression.

$$RSS_{\text{ridge}} = RSS + \lambda\acute{\delta}^T\acute{\delta} \quad (7)$$

High estimations of may lead to a high sum-of-squares error and lead to the rejection of the estimate. In order to obtain the covariance matrix after inverting XT X, ridge regression is often used. Here, the ridge coefficient promotes non-singularity by perturbing the diagonal elements of X X.

Thus, RR is used in multicollinearity-affected multiple regression analyses. The cave allows L.S. to make objective judgements. Ridge regression is used to lower standard errors because regression estimates are biassed in a certain way.

4.Results and Discussions

4.1 Implementation Setup

The results of experiments testing the predicted FF-RR method's performance are shown below. The goal of this effort was to improve clustering quality and algorithmic parallelism via estimation. Each of the 10 nodes in the Perl and Hadoop cluster had a CPU speed of 2.26 GHz, 2 GB of RAM, and a 120 GB hard drive, and the experiments were run on this setup. Each node runs Ubuntu 14.04 with Apache Hadoop 2.6.2.6.5 as the operating system. The experiment is broken up into two parts so that we may get a better idea of how well our FF-RR approach performs on a large-scale dataset. To begin, we use the simulated data sets shown in Table 1. Using the UCI Machine Learning Repository's baseline datasets as a starting point, we synthesised data that includes a rich collection of characteristics, measured in terms of both the number of dimensions and the number of clusters. Duplicate records were produced for each baseline dataset around 106 times, leading to massive dataset sizes of about 107 records. The second step of a disc drive production process involves implementing the predicted method on the fabricated data set.

Table 1. Dataset description

Dataset	#Records	#Dimensions	Clusters
I	10,000,050	2	3
II	10,000,197	1	8

III	10,000,040	1	12
IV	10,000,822	5	2

F-Measure is a measure from the field of information retrieval that estimates clustering accuracy using the concepts of accuracy and recall [19].

$$F(l,j) = \frac{2-r(L,j)p(L,J)}{r} \qquad (8)$$

Where r and p indicates recall and precision.

The general F-Measure for the whole dataset of size is acquired in Eq. (9):

$$F = \sum_i \quad n_i/n_j\,(F(i,j)) \qquad (9)$$

Where F has an upper limit rate of 1.0, which occurs when cluster j acquires only Class i objects for all Class i objects. Evidently, the F-Measure is more pronounced in high-quality cluster solutions. We first compared the FF-RR method's results to those of the PK-Means algorithm [8] and the parallel K-PSO algorithm [9] to get a feel for their impact on cluster quality. The quality of the clustering results obtained by the algorithms is shown in Table 2 for each dataset. For first cluster centroid estimation, a random sample of 0.1% of data records is used.

Table 2.F=Measure outcomes gained by the algorithms on the synthetic dataset

Dataset	**FF-RR**	**PK-Means**	**Parallel K-PSO**
I	0.942	0.667	0.785
II	0.487	0.298	0.324
III	0.818	0.482	0.517
IV	0.743	0.586	0.627

As can be seen in Table 2, our method achieves rather acceptable results across the board with this dataset. When compared to PK-Means and the K.PSO parallel approaches, the F-measure values achieved by the proposed algorithm FF-RR are higher. Therefore, the envisioned procedure yields improved quantitative assessment results. To calculate the speed up Eq. (10):

$$S_p = \frac{T}{T_n} \qquad (10)$$

Where T is the time, it takes for one node's code to run, TN is the total time it takes for N nodes to run their code.

Figures 2 and 3 show the F-Measure results over the Synthetic dataset acquired by the methods for datasets I, II, III, and IV, respectively. Figures 4 through 7 show the execution timings and optimisation techniques.

The execution time of FF-RR in the Hadoop cluster decreases linearly with an increase in the number of nodes, as shown in Figures 2 and 3. Cluster performance is worse for the dataset with the fewest number of dimensions.

Table 3 illustrates that the efficiency results in big dataset volumes with 10 nodes collected by MRFF are focused on performance scalability across the dataset sizes. Eq. (10) might be used to get an estimate of the effectiveness. Keep in mind that for each dataset, we replicate the records multiple times, increasing the size from 2 GB to 10 GB.

$$\text{Efficiency} = \frac{\Sigma_P}{N} \qquad (11)$$

Table 3. The effects of the FF-RR algorithm's efficiency on the fluctuation of dataset volumes with 10 nodes across dataset I

	Dataset sizes (GB)				
	1	3	5	7	9
F-Measure	0.942	0.942	0.942	0.942	0.942
Efficiency	0.955	0.964	0.968	0.972	0.975

Table 4. Variation in dataset volumes obtained by the FF-RR technique with 10 nodes, and the implications for efficiency

	Dataset sizes (GB)				
	1	3	5	7	9
F-Measure	0.942	0.942	0.942	0.942	0.942
Efficiency	0.955	0.964	0.968	0.972	0.975

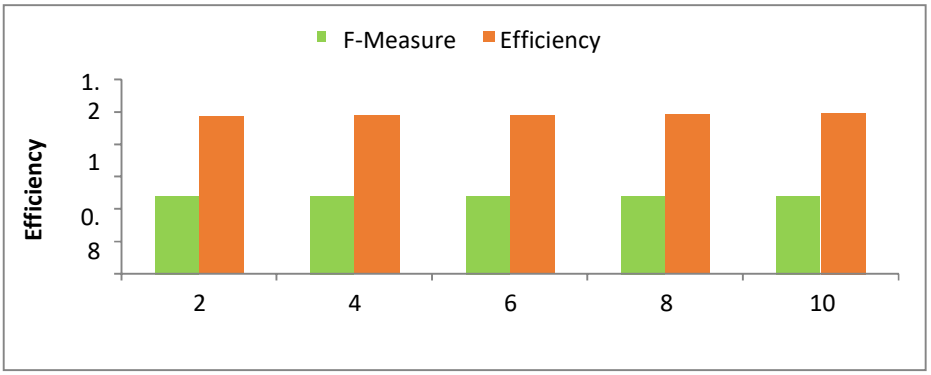

Figure 3: Results of FF-RR's execution on 10 nodes in a Hadoop cluster, broken down by dataset I's execution time and speed-up.

Table 5 Comparison of the FF-RR algorithm's efficiency in acquiring datasets with 10 nodes to dataset II

	Dataset sizes (GB)				
	1	3	5	7	9
F-Measure	0.376	0.389	0.256	0.789	0.578
Efficiency	0.863	0.894	0.753	0.894	0.996

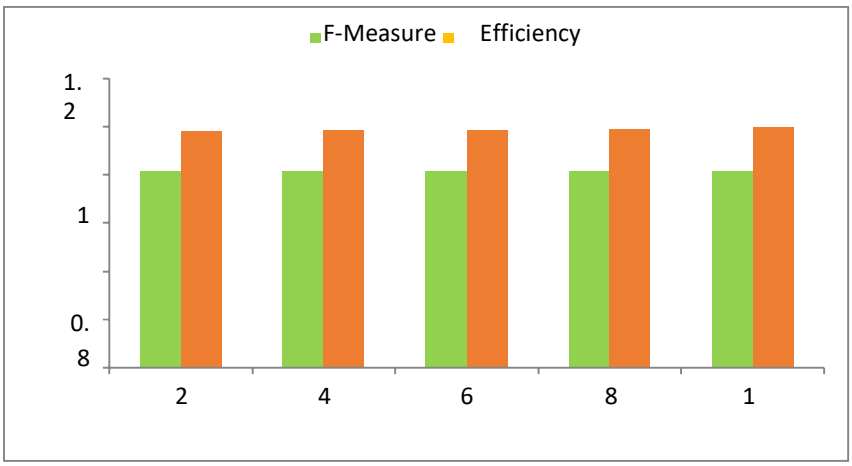

Figure 4: Dataset II-specific performance metrics for FF-RR's execution on a Hadoop cluster of 10 nodes

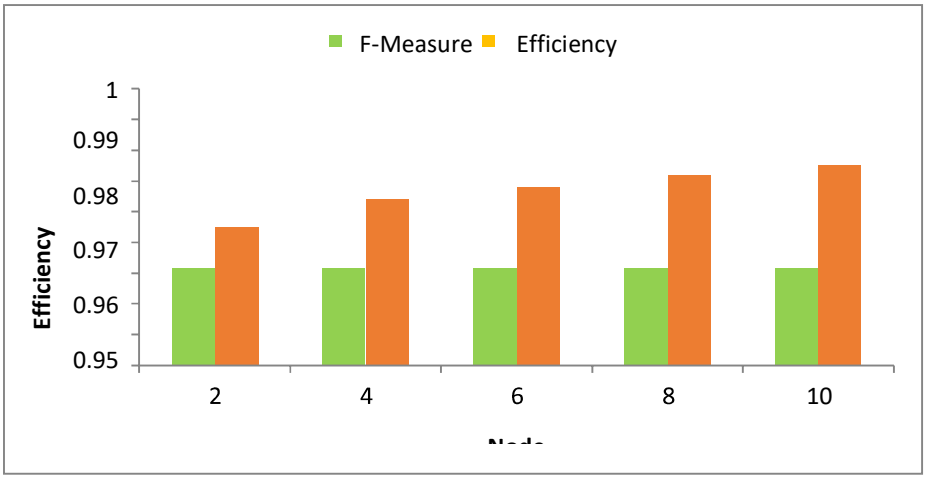

Figure 5: Results of running FF-RR on the dataset III with ten nodes in a Hadoop cluster in terms of execution time and speedup

Table 6. The effects of volume variation in datasets obtained using the FF-RR technique with 10 nodes dataset III

	Dataset sizes (GB)				
	1	3	5	7	9
F-Measure	0.717	0.717	0.717	0.717	0.717
Efficiency	0.972	0.974	0.974	0.992	0.997

The K-Means method is quite quick when it comes to summarising experimental results in the issue area. However, it is very sensitive to changes in the position of the cluster's original centre. The K-Means function's solution often converges to a set of local optimum values close to the search's origin. In a direct comparison with PSO, the FF method can yield better outcomes. While PSO simply applies an exploitation-dependent adjustment to particle velocities, the FF approach makes use of both exploitation and exploration throughout the search process. The FF method's brighter FFs are responsible for managing exploitation while the method's darker FFs are responsible for handling exploration. When various solutions are mixed up at any local optimum, the FF will attempt to locate a new one at random. In addition, the time and money needed for a big data clustering method are reduced thanks to this implementation of Map-Reduce with the FF algorithm. The results suggest it is an excellent choice for analysing vast amounts of data due to the distributed algorithm's ability to preserve cluster quality.

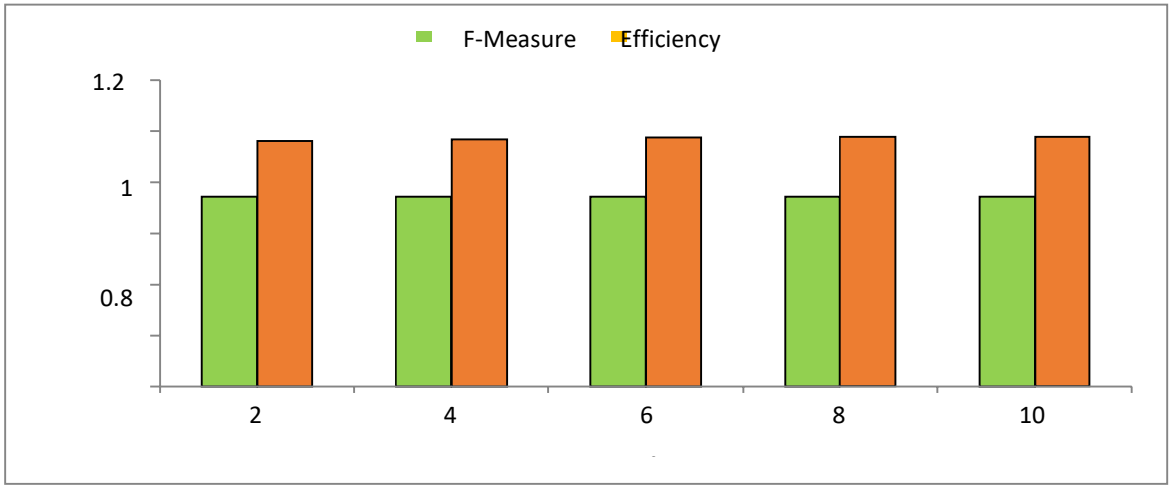

Figure 6: Time required for FF-RR to complete and performance increase while using 10 nodes in a Hadoop cluster on the dataset IV

The performance of the FF-RR approach is shown to improve as the size of the dataset grows enormously in Tables 4 through 6. In all other respects, the effectiveness was 90%. As the issue volume grows, the algorithm can keep up with it. The second part makes use of a very enormous dataset with information gleaned from the hard disc drive manufacturing process, including things like the materials used, the instruments used, and the procedures carried out. The effectiveness and efficiency of the FF-RR method are evaluated and calculated. There are two sets of numbers here: those for which the product objects passed the test (1,457,000 records) and those for which they did not (143,000 records). There are 30 distinguishing features used to classify these categories, including qualities and machine parameters. The Figures 4-7 data set was used to measure the execution time and speedup results of FF-RR using a cluster of 10 nodes in Hadoop. Again, FF-RR outperforms other algorithms when it comes to fixing this problem. Efficiency Nodes Effectiveness on the F-Scale. However, when compared to other methods, the FF-RR methodology delivers a 58% improvement in the standard F-measure.

5.Conclusion

Here, we focus using an FF-RR technique to do the clustering. It runs on the Hadoop framework and employs the Map-Reduce paradigm. The results of the experiments show that the proposed method performs better while working with large datasets, and that the quality of the clustering is preserved. This paper proposes FF-RR, a Map-Reduce-based FF technique, to cluster large datasets. The clustering process for massive datasets is enhanced by experimenting with the FF algorithm inside the Map-Reduce framework in the Hadoop environment. This paper also includes a comparison of current approaches like PK-Means and parallel K-PSO, with a focus on the former.

5.1 Future Work

It follows that the anticipated method is superior in both quality and efficiency. As such, it provides a viable alternative in situations when large amounts of data need to be grouped together.

References

[1] Bae, M., Yeo, S., Park, G., & Oh, S. (2021). Novel data-placement scheme for improving the data locality of Hadoop in heterogeneous environments. Concurrency and Computation: Practice and Experience, 33(18), e5752.

[2] Bae, M., Yeo, S., Park, G., & Oh, S. (2021). Novel data-placement scheme for improving the data locality of Hadoop in heterogeneous environments. Concurrency and Computation: Practice and Experience, 33(18), e5752.

[3] Rajeh, W. (2022). Hadoop distributed file system security challenges and examination of unauthorized access issue. Journal of Information Security, 13(2), 23-42.

[4] Ding, F., & Ma, M. (2023). Data locality-aware and QoS-aware dynamic cloud workflow scheduling in Hadoop for heterogeneous environment. International Journal of Web and Grid Services, 19(1), 113-135.

[5] Fu, Z., He, M., Tang, Z., & Zhang, Y. (2023). Optimizing data locality by executor allocation in spark computing environment. Computer Science and Information Systems, 20(1), 491-512.

[6] Fazul, R. W. A., & Barcelos, P. P. (2022, April). An event-driven strategy for reactive replica balancing on apache hadoop distributed file system. In Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing (pp. 255-263).

[7] Khan, M., Liu, Y., & Li, M. (2014, August). Data locality in Hadoop cluster systems. In 2014 11th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD) (pp. 720-724). IEEE.

[8] Lee, S., Jo, J. Y., & Kim, Y. (2018, December). Key based Deep Data Locality on Hadoop. In 2018 IEEE International Conference on Big Data (Big Data) (pp. 3889-3898). IEEE.

[9] Lee, S., Jo, J. Y., & Kim, Y. (2019, May). Survey of data locality in apache hadoop. In 2019 IEEE International Conference on Big Data, Cloud Computing, Data Science & Engineering (BCD) (pp. 46-53). IEEE.

[10] Moses, T., & Abiodun, O. J. (2023). A fault-tolerance model for Hadoop rack-aware resource management system. Journal of Computer Science and Engineering (JCSE), 4(1), 15-24.

[11] Oliveira, F., Alves, A., Moço, H., Monteiro, J., Oliveira, Ó., Carneiro, D., & Novais, P. (2023). Dynamic Management of Distributed Machine Learning Projects. In Intelligent Distributed Computing XV (pp. 23-32). Cham: Springer International Publishing.

[12] Sharma, A., & Singh, G. (2018, December). A review on data locality in hadoop MapReduce. In 2018 Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC) (pp. 723-728). IEEE.

[13] Sharma, S., & Bharti, R. K. (2023). New efficient Hadoop scheduler: Generalized particle swarm optimization and simulated annealing-dominant resource fairness. Concurrency and Computation: Practice and Experience, 35(4), e7528.

[14] Thu, M. P., Nwe, K. M., & Aye, K. N. (2019). Replication based on data locality for hadoop distributed file system (Doctoral dissertation, MERAL Portal).

[15] Thu, M. P., Nwe, K. M., & Aye, K. N. (2019). Replication based on data locality for hadoop distributed file system (Doctoral dissertation, MERAL Portal).

[16] Xia, L., Sun, W., Liu, X., Sun, G., & Jiang, X. (2023, February). Blaze: A High-performance Big Data Computing System for High Energy Physics. In Journal of Physics: Conference Series (Vol. 2438, No. 1, p. 012012). IOP Publishing.

[17] Yılmaz, S & Küçüksille, EU 2015, 'A new modification approach on bat algorithm for solving optimization problems', Applied Soft Computing, vol. 28, pp. 259-275.

[18] Yoan Miche, Anton Akusok, David Veganzones, Kaj-Mikael Björk, Eric Séverin, Philippe du Jardin, Maite Termenon, and Amaury Lendasse. 'SOM-ELM—Self-Organized Clustering using ELM.'Neurocomputing, vol.165, pp.238-254, 2015.

[19] Yongjiao Sun, Ye Yuan, and Guoren Wang. 'An OS-ELM based distributed ensemble classification framework in P2P networks.' Neurocomputing, vol.74, no.1, pp.2438-2443, 2011.

[20] Nalajala, A., Ragunathan, T., Rajendra, S. H. T., Nikhith, N. V. S., & Gopisetty, R. (2019, July). Improving performance of distributed file system through frequent block access pattern-based prefetching algorithm. In 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT) (pp. 1-7). IEEE.

[21] Kazi, A. N., & Chaudhari, D. N. (2022). A HADOOP ANALYSIS OF MAPREDUCE SCHEDULING ALGORITHMS. Specialusis Ugdymas, 1(43), 7345-7350.

[22] Benlachmi, Y., El Yazidi, A., & Hasnaoui, M. L. (2021). A comparative analysis of hadoop and spark frameworks using word count algorithm. International Journal of Advanced Computer Science and Applications, 12(4), 778-788.

[23] Yuan F, Lian F, Xu X, & Ji Z, Decision tree algorithm optimization research based on MapReduce, Software Engineering and Service Science, 6th IEEE Int Conf in 2015 1010-1013.

[24] Zaki, MJ & Hsiao, C-J 2002, 'CHARM: An efficient algorithm for closed itemset mining', in Proceedings of the 2002 SIAM international conference on data mining, pp. 457-473.

[25] Cholissodin, I., & Supianto, A. A. (2019, September). Enhancement Full Open Source Hadoop Distribution Universal Big Data Up Projects (UBig) From Education To Enterprise. In 2019 International Conference on Sustainable Information Engineering and Technology (SIET) (pp. 90-93). IEEE.

[26] Yulong, Z., & Weiting, L. (2020, October). A research on battlefield situation analysis and decision-making modeling based on a Hadoop framework. In 2020 2nd International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI) (pp. 388-391). IEEE.

[27] Zhang, J., & Lin, M. (2023). A comprehensive bibliometric analysis of Apache Hadoop from 2008 to 2020. International Journal of Intelligent Computing and Cybernetics, 16(1), 99-120.

[28] Zhang, P., Li, C., & Zhao, Y. (2016, December). 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.

[29] Adhikari, B. K., Zuo, W., Maharjan, R., Han, X., Amatya, P. B., & Ali, W. (2019, August). Statistical analysis for detection of sensitive data using hadoop clusters. In 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS) (pp. 2373-2378). IEEE.

[30] Jadhav, A. D., & Pellakuri, V. (2021). Highly accurate and efficient two phase-intrusion detection system (TP-IDS) using distributed processing of HADOOP and machine learning techniques. *Journal of Big Data*, *8*(1), 1-22.