CHAPTER 1
INTRODUCTION

Importance of clustering can be gauged by the fact that it is actively being used in domains like data mining, image segmentation and information retrieval [1]. More importantly it is an unsupervised learning task which makes it better than supervised learning techniques like classification algorithms which analyses class-labels and requires training. Most of the online data is generated on social media platforms which do not have a predefined model, so clustering being an unsupervised mechanism has a distinct advantage.

The global data generation trends are shooting up really fast and CPU clock speeds have almost hit their limits, the fastest being 8.805 GHz and 28,875 GB of data generated per second way back in 2013. [2] These trends route us to parallelization methods. On similar lines traditional clustering algorithms have high computation time due to I/O's and data analysis. There are two problems inherent to using most clustering algorithms to process very large databases. The first problem is determining how to process the data without having it all in RAM at one time, and the second problem is how to complete all the computation in a reasonable amount of time. Metaheuristic algorithms provide a possible solution to deal with high computation time of clustering. Nature inspired metaheuristic algorithms are simple to implement and are highly efficient in finding global optimum. An important consideration however is solution diversity and solution speed. Increase in diversity can be achieved by randomization and stochastic intervention, implementing any one or both of which implies higher computation time. There always has to be trade-off between optimum solution and computation time. In addition to metaheuristic algorithms, various parallelization approaches can also be plugged in to reduce computation time and at the same time make it scalable and further reduce processing time. This approach can be used to curb the trade-off as higher number of iteration can be achieved in in the same time as compared to sequential metaheuristic algorithms.

MapReduce architecture has been successfully implemented and has been successful to deal with large data sets which need parallel processing [3]. A MapReduce program is composed of a Map procedure (method) that performs filtering and sorting (such as sorting students by first name into queues, one queue for each name) and a Reduce method that performs a summary operation (such as counting the number of students in each queue, yielding name frequencies). The "MapReduce System" (also called "infrastructure" or "framework") orchestrates the processing by marshalling the distributed servers, running the various tasks in parallel, managing all communications and data transfers between the various parts of the system, and providing for redundancy and fault tolerance.

1.1 MOTIVATION OF THE WORK

Partitioning clustering algorithms like K-means and Clustering Large Applications based on Randomized Search -CLARANS are easy to scale as compared to hierarchical and density based clustering which is a suitable trait, as size of dataset increase these algorithms would scale up provided parallelization of these are also in place to reduce computation time. Even after all the computation K-means is likely to get stuck into some local optima and even after many iteration is likely to remain so. There have been other optimizations carried out with K-means like PSO with K-means. But PSO has certain disadvantages like slow convergence in refined search stage and weak local search ability. Also the PSO search is not applicable for problems involving non-coordinate systems like the solution of energy field and the moving rules for the particles in the energy field. Parallel black hole speeds up computation and finds better results for better sized data. The algorithm has been implemented in Apache Hadoop MapReduce architecture. Also black hole avoids the local optimum problem by generating a random star which is distant from the optimum which has been currently calculated. This allows it to make even diverse searches on data sets and hence avoid any bias in a given direction. K-means with ACO is efficient in parallel form as it is guaranteed to converge towards global optimum but it has a disadvantage that the time taken to converge is uncertain and dependent on type of data set. Parallel Black Hole algorithm unlike K-means is parameter free which means there will have be no parameters to be set initially which further betters the algorithm.It also improves the quality of solution found by either k-means or other variations. The algorithm’s convergence speed is also high Like sequential K-means, black hole also takes a lot of time in fitness computation and updating the particle current location. So the further work implements and studies one such partition cluster algorithm- Parallel Black Hole algorithm.

1.2 AIM OF THE THESIS

The objectives of the thesis is to

- To realize an approach to cluster the data efficiently and to speed up the entire process by the virtue of MapReduce programming model of hadoop.
- To improve the strategy of initial selection of centroids.
- To completely eradicate the problem of local-minima by effectuating principle of global optimization of the meta –heuristic algorithms
- To determine which data belongs to which cluster by the use of weight matrix.
- By enhancing the overall performance of data clustering by employing more number of machines in hadoop distributed system instead of increasing RAM and CPU for scaling up which is more inconvenient.
- To check the performance of the proposed algorithm on different input datasets and comparing their execution time.
- To develop a comparative study of the proposed algorithm and the existing algorithms.

1.3 ORGANISATION OF THE THESIS

The thesis in all consists of seven chapters and references

Chapter 1 is the introduction. It evokes the motivation of the work, aim of thesis and structure of thesis.

Chapter 2 description of work done by various people and their contribution. It also explains what clustering is and types of data clustering algorithms. It also reports the different types of the existing data clustering algorithms.

Chapter 3 Introduction and explanation of hadoop.

Chapter 4 This chapter discusses the sequential Algorithm which was taken as an inspiration for the research design and architecture of the proposed integrated black hole clustering MapReduce algorithm

Chapter 5 Experimental results

In the end, bibliography is being included .

CHAPTER 2

LITERATURE SURVEY

2.1 META-HEURISTIC ALGORITHMS

In the present scenario nature inspired meta-heuristic algorithms are becoming very powerful in solving problems based on optimizations [4]. Heuristic in meta-heuristic term means discovering by trial and error and 'meta' means on a higher level. These algorithms involve randomization which helps in searching on a global scale rather than local search as done in conventional algorithms.

Hence meta-heuristic algorithms are suitable for the applications where global optimization is required [5] [6]. Producing acceptable solutions in reasonable time is also one of the key features of Meta -heuristic algorithms. [7] Resources like memory have always been a concern, an algorithm is efficient if it uses the resources in an optimized way and always maintains a trade-off between time and space complexity.

2.2 TYPES OF META- HEURISTIC ALGORITHMS

There are a number of nature inspired algorithms, enumerating few are,

1.The ant colony optimization (ACO) [8]: It is influenced from the behaviour of real ant as it finds the shortest path between the food and the nest. It is a swarm based algorithm. In real life ants walk randomly in search of food, if an ant finds food then ant returns back to the colony by leaving trails of pheromone. Now all the other ants follow this route depicted by pheromone imprints. Now the ants stop going through random routes all the ants now follow the trails laid by that ant to find food. One of the major constraint is that as time passes the pheromone evaporatesand thus decreasing its attaching strength.Thus this forces ant to find the shortest path so that pheromone trails are not evaporated and time reduces. Pheromone trails evaporation has an advantage of not getting stuck in local optima. An ant represents a potential solution, thus objective function operated on ant pheromone density is related to each associated route of food and its evaporation is due to premature solution stagnation.

2. The gravitational search algorithm (GSA): It is inspired by a physical phenomenon and is based on the Newtonian gravity and the laws of motion. GSA is combined with kmeans algorithm to form a hybrid algo. [9] In this approach first step is selection of initial centroids by k-means and produces near about optimal ceneters of the clusters. In the second step, an initial population of solutions is generated ,which will be applied by the GSA algorithm in the third step.

3. The particle swarm optimization (PSO) : It is an evolutionary technique for optimizing functions designed based on behavior of swarms of birds, fishes etc. In PSO, there are particles, having variables of an optimization problem,and they are scattered in the search environment. Some particles have better positions than others and they are termed as personal best and global best in consecutive iterations. [10] Therefore, based on aggregative particles' behavior, other particles may try to raise their position to the prior particles' positions and become global best.

4. The BAT algorithm [11]**:** It is based on the echolocation behaviour of bats. The bats echolocation property helps them to locate their prey even in complete darkness.

5. The Cuckoo Search Algorithm: In the Cuckoo search algorithm, [12] every egg depicts a solution and a cuckoo egg is new solution, the purpose is to use the new and better solutions to replace a poor solution in the nest. The next generation only carries the best nests having eggs of high quality.

6. The Genetic algorithm: The genetic algorithms are based on biological evolution . Each population member is known as a candidate solution and has its characteristics which can be mutated and altered. [13] Conventionally the solutions are represented as strings of 0's and 1's. After the representation and the fitness function are consolidated , the genetic algorithm proceeds to initialize a population of solutions and then improvement is done by repetitive application of the mutation, crossover, inversion and selection operators. The termination occurs if the solution satisfies minimum criteria or maximum number of iterations have reached.

7. Fire Fly Algorithm: The fire fly algorithm [14] which is based on their major characteristics of flashing of lights. It is a popular algorithm used to solve diverse problems. Bioluminescence is the biochemical process by which lights flash. Such flashing light may serve as the primary courtship signals for mating.

8. Big Bang crunch algorithm: The BB-BC algorithm comprises two phases in the Big Bang phase, energy dissipation leads to disorder and randomness and it is the main feature of this phase; whereas, in the Big Crunch phase, randomly distributed particles are drawn into an order. Inspired by this theory, Big Bang–Big Crunch (BB–BC) [15] method generates random points in the Big Bang phase and shrinks those points to a single representative point via a center of mass or minimal cost approach in the Big Crunch phase.

Meta-heuristic algorithms prove to be efficient in many applications as they provide global optima ,but the major drawback of these algorithms is that they are very computation intensive.
Hence distributed architectures came into picture. MapReduce is one of the distributed data processing framework which is used nowadays to increase the efficiency of the clustering algorithms.Few of the above meta-heuristic algorithms are combined with K-means clustering algorithm and are implemented on MapReduce are discussed below-

1. K-means with PSO [16]: K-PSO combines Particle Swarm Optimization (PSO) with K-means based on MapReduce.It takes advantage of PSO to improve the global search ability of K-means, and then it makes K-means parallel with MapReduce to enhance its capability of processing massive data.
2 **K-means with genetic algorithms on MapReduce**- The genetic algorithms are difficult to parallelize due to there sequential behaviour . In this algorithm the mapper does the population initialization, each individual of population is a chromosome of size N. Each segment of chromosome is a centroid. After that fitness values were calculated and in the second phase the reducer form a new chromosome by joining the results received from the mapper. Process is repeated until all the centroids of the chromosomes have an inter cluster value greater than the threshold value and the final chromosome contains the location of the optimal clusters.

3. K-means with ACO on MapReduce: the identification of species from its genome sequence is done in this approach.Feature descriptors for a genome sequence are identified using

MapReduce on Hadoop framework. [17] Each feature descriptor is a three lettered keyword generated using A, T, C, G nucleotide bases. Genome sequences of related species are clustered by considering the feature descriptor count. MapReduce version of clustering model that uses K-means, Differential Evolution(DE) and Ant Colony Optimization(ACO) has been proposed. This MapReduce model improves accuracy as the entire genome sequence is considered. The inherent parallelism in the MapReduce model also enhances execution time efficiency.

4. K-means with gravitational search on MapReduce [18]: The GSA-KM algorithm helps the k-means algorithm to escape from local optima and also increases the convergence speed of the GSA algorithm.

The main motivation behind proposing the ‘Black Hole MapReduce’ algorithm is that the existing algorithms need to set the parameters manually also when large datasets are set as input the time taken was really substantial. Following chapters explain the intricacies.

CHAPTER 3

RESEARCH METHODOLOGY

3.1 APACHE HADOOP

Apache hadoop is a license free software framework for storing huge datasets and processing them with the use of clusters of commodity hardware i.e. inexpensive hardware. It is influenced from Google File System (GFS) and Google's MapReduce and allows the applications to work with thousands of nodes i.e. commodity hardware and petabytes of data.

Apache hadoop [19] is framework that supports data intensive applications. The concept of MapReduce is used to process the data stored in HDFS. The HDFS is a fault tolerant file system which allows to replicate the data to multiple data nodes to prevent any sort of data loss. [20] It also provides high throughput to access the data Hadoop is supported by a various operating systems such as CentOS, SLES ,RHEL , Oracle Linux with default kernel, Ubuntu , MAC OS, and OpenSolaris.

3.1.1 KEY FEATURES OF HADOOP

- **Hadoop processes data faster:** hadoop performs exceptionally well when processing high volume data as compared to the traditional batch processing systems which take hours to just load the real-time data. It outperforms all the existing mainframes as it processes extremely well.
- **Flexible:** There is a severe lack of technology to analyze unstructured data due to which it is often ignored hence only 20% of the data in an organization is structured and therefore is used in decision-making. Hadoop is capable of processing any sort of data whether it is encoded or formatted , structured or unstructured. It is capable of making decision with all sorts of data.
- **Fault Tolerant:** Failures are very common when we are working real-time, large number of machines. Hadoop copes up with them by the concept of replication. It is an extremely reliable storage system as the level of replication is configurable i.e. one can manipulate the number of copies to be stored on the nodes, hence the system can never

go out of service. It also reallocates the work if the node is down.

- **Scalable:** As the data volume is increasing exponentially the system must be capable of bear the changes. It should be able to grow without altering the existing system and programs. Any number of nodes can be added to the existing distributed system expanding the storage and processing of the overall system.
- **Robust:** From developers to small or large scale organizations hadoop’s ecosystem meets their analytical needs. To enumerate few of the modules are Hadoop YARN/MapReduce, Hive ,Pig, HBase, Sqoop, Zookeeper, Avro, Cassandra, Mahout etc.
- **Cost effective:** A substantial amount of cost reduce, as massive parallel computing is done on cheap commodity hardware generating costs benefits. Hadoop data management system expenses approximately comes out to be one-fifth of the one –twentieth cost of other data management systems.

3.1.2 HADOOP CLUSTER

Hadoop Cluster is a set of "cheap" commodity hardware networked together which resides in the same location i.e. set of servers resides in set of racks which are in data centre. “Cheap” Commodity Server Hardware means that there is no need for super-computers, and can use commodity unreliable hardware. The hardwares used are not desktops but servers. Hadoop Cluster is a collection of Hadoop nodes where each node consists of a Processor and Storage as shown in figure 1. In Hadoop cluster, processors access underlying local storage and execute code.

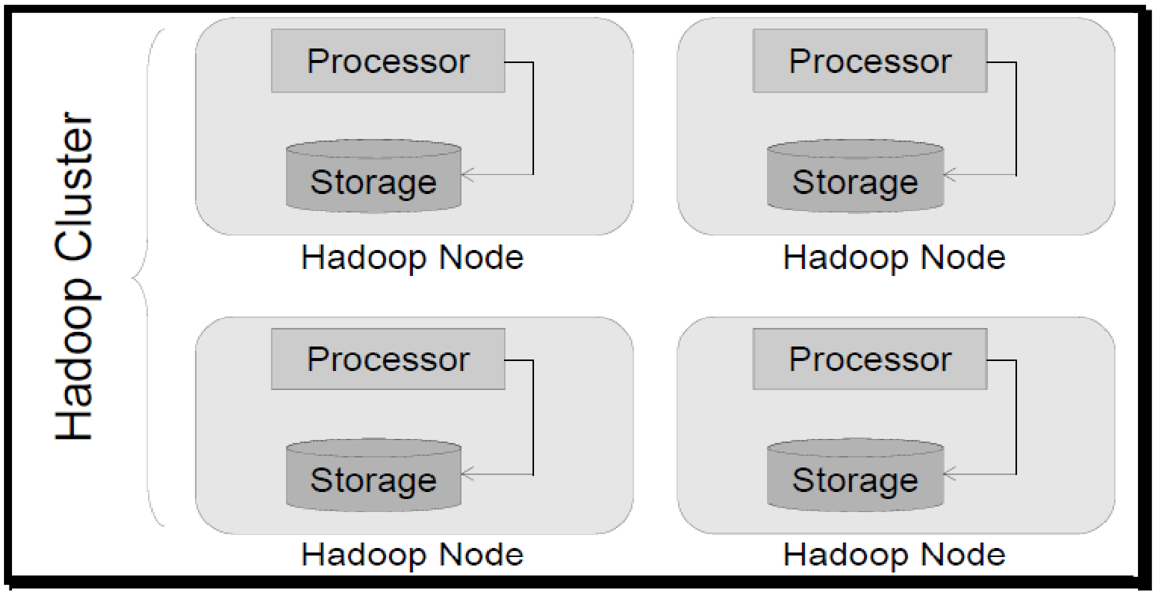

Figure 1 Hadoop Cluster [23]

3.1.3 Why Hadoop?

Hadoop in the cloud is an upcoming topic. This section explains six reasons why this association of Hadoop and Cloud makes sense, and why customers are seeing advantage in this model. The reasons are :

1. Lowering the cost of innovation

Running Hadoop on the cloud makes sense similarly as running any other software on the cloud. The cloud also makes sense for a quick and one time use case involving big data computation.

2. Acquiring large scale resources quickly

Hadoop and the platforms it was inspired from made the vision of compute and linear storage using commodity hardware a reality. Internet giants for example Google invested in building this hardware themselves.

In the enterprise Hadoop adoption grew from tens of nodes to a large or medium-sized cluster with a few hundreds of nodes. These clusters are managed by a team which is different from the Infrastructure team and is called „data platform‟ team. With increasing Analytics, need to increase capacity of Hadoop clusters also grew. It was proved that increasing more hardware in Hadoop cluster wasn‟t as easy or fast as it should be.

3. Efficiently Handling Batch Workloads

Hadoop being batch oriented has usage patterns involving scheduled jobs processing new incoming data on a temporal or fixed basis. Activity data from web server logs and devices is being collected by companies and ingested for analysis through an application on Hadoop. The load on resources of a Hadoop cluster varies based on rate of incoming data or on the timings of these scheduled runs. The cloud is more efficient to handle such batch workloads with its pay as you use model.

4. Handling Variable Resource Requirements

All Hadoop jobs are not equal. Some of them require more memory while some require more compute resources, and some others require a lot of I/O bandwidth. Usually, a physical Hadoop cluster is built of similar machines that are large enough to handle the largest job. For example a job can affect tasks of other jobs if its task requires more memory than average due to a drain on system resources.

Cloud solutions already offer end user a choice to provision clusters with different machines for different workloads.

5. Running Closer to the Data

Data starts living on the cloud as businesses move their services to the cloud. Analytics thrives on large volumes data so analytical platforms i.e. Hadoop clusters should exist on the cloud or in same cluster environment.

6. Simplifying Hadoop Operations As cluster association happens the isolation of resources gets lost for different sets of users. Multi-tenancy issues like varied security constraints, user jobs interfering with one another etc. arises as all user jobs get bunched up in a shared cluster.

This can be resolved using cluster level policies that prevent users from doing anything harmful to other user jobs.

Using the cloud, user can run different types of clusters with different configurations and characteristics, each suitable for a particular set of jobs.

3.1.4 Hadoop Distributed File System (HDFS)

Hadoop Distributed File System is a file system that runs on top of native file system like Ext3, Ext4 and others, and is based on Google file system. It gives user appearance of a single disk. It is highly fault tolerant in a way that it can handle disk crashes, machine crashes, etc. It is built upon cheap commodity hardware which reduces the overall cost of installation of Hadoop [21].

3.1.4.1 Where HDFS should be used?

1. Hadoop Distributed File System is good for storing large files, managing storage of Terabytes, Petabytes and more, millions rather than billions of files storage and 100MB or more data per file in storage.

2. HDFS supports Streaming data in which pattern is write once and read-many times. HDFS is optimized for streaming reads. A new feature of appending an existing record in HDFS has been added in versions after 0.21 of Hadoop.

3. HDFS uses cheap commodity hardware which are less reliable which implies that it does not require super computers to work on.

3.1.4.2 Where HDFS should not be used?

1. HDFS is not good for high throughput instead should be used for low-latency for small chunks of data.
2. It is good for millions of large files rather than billions of small chunks, for example each file can be 64 MB or more.
3. It does not support multiple writers – single writer per file is supported. It only appends data i.e. writes only at the end of file, no-support for writing at arbitrary offset like in Google file system.

3.1.4.3 HDFS Daemons

File system cluster is being managed by three types of processes namely, Namenode, Datanode and Secondary Namenode [22].

- **Namenode:** It manages the file systems namespace, meta-data and file blocks. It runs on one machine and manages several machines. All datanodes report to Namenode about their presence and according to the number of available datanodes it manages degree of replication as decided by the Administrator. For fast access Namenode keeps all block meta-data in memory. The other role is to serve the client queries, it allows clients to add/copy/move/delete a file, it will records the actions into a transaction log. For the performance, it save the whole file structure tree in RAM and hard drive. A HDFS only allow one running namenode, that's why it is a single point of failure, if the namenode failed or goes down, the whole file system will goes offline too. So, for the namenode machine, we need to take special cares on it, such as adding more RAM to it, this will increase the file system capacity, and do not make it as DataNode, JobTracker and other optional roles.

- **Datanode:** It stores and retrieves data blocks according to the request after it has reported to Namenode about its health. It runs on many machines and forms the cluster. On startup, DataNode will connect to the NameNode and get ready to respond to the operations from NameNode. After the NameNode telling the position of a file to the client, the client will directly talk to the DataNode to access the files. DataNodes could also talk to each other when they replicating data. The DataNode will also periodically send a report of all existing blocks to the NameNode and validates the data block checksums.
- **Secondary Namenode:** It performs the house keeping work so that Namenode doesn‟t have to do it and reduces the load of Namenode. It requires similar hardware as Namenode machine and is not used for high-availability – not a backup for Namenode. Its works is to back-up the metadata and store it to the hard disk, this may helping to reduce the restarting time of NameNode. In HDFS, the recent actions on HDFS will be stored in to a file called EditLog on the NameNode, after restarting HDFS; the NameNode will replay according to the Editlog. Secondary NameNode will periodically combines the content of EditLog into a checkpoint and clear the Editlog File, after that, the NameNode will replay start from the latest checkpoint, the restarting time of NameNode will be reduced.

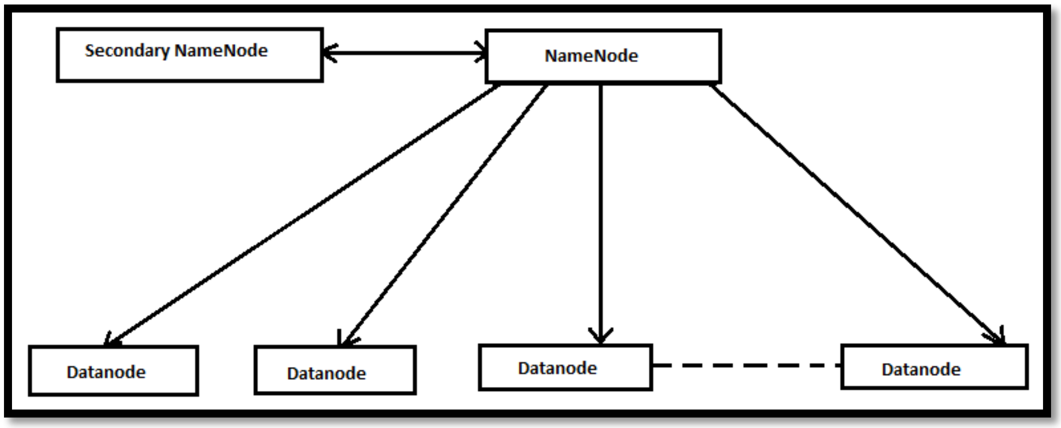

Figure 2 Processes of HDFS

3.1.5 Files and Blocks

Files on HDFS are stored after splitting them into blocks. A block is a single unit of storage. They are managed by Namenode and stored by Datanode. This process of storage and management is transparent to the user. The blocks are replicated across multiple machines at load time as shown in figure 8 i.e. same block is stored on several machines which is good for fault-tolerance and quick access which improves response time. The default replication factor is 3 but can be changed as required.

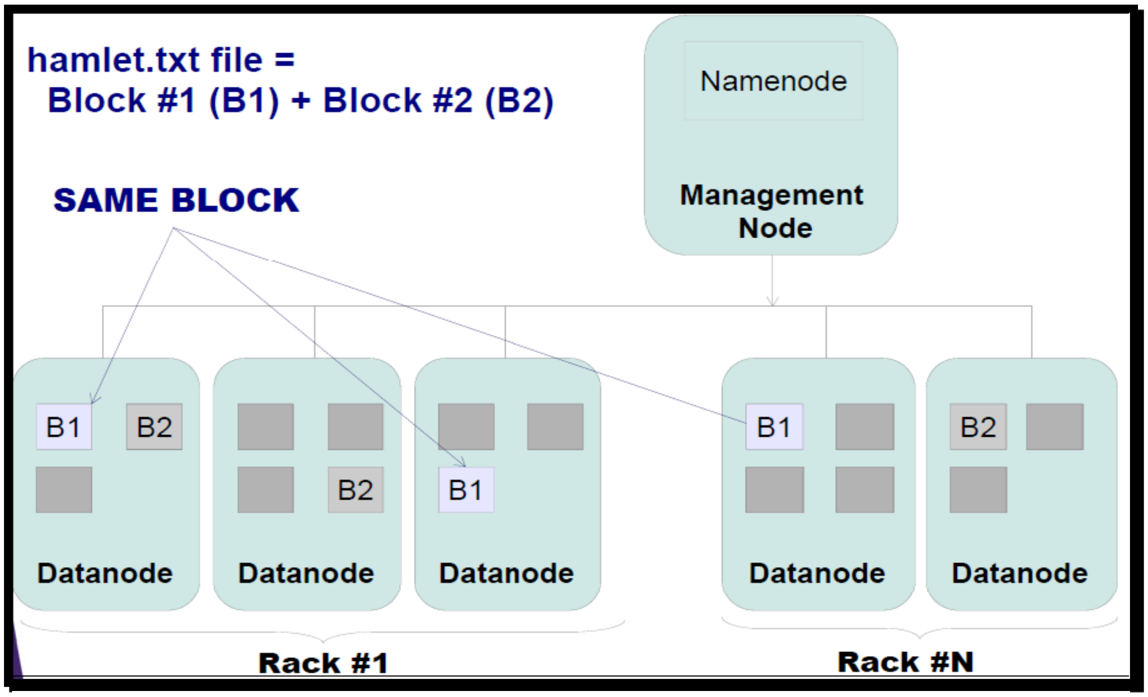

Figure 3 Replication of Blocks with replication factor 3 [21]

Blocks are normally either 64 MB or 128 MB but default size is 64 MB. The motivation of using blocks is to minimize the cost of seek as compared to transfer rate i.e. Time to seek should be less than Time to transfer. Namenode determines replica placement in datanode i.e. replica placements are rack aware. This process attempts to reduce bandwidth and improves reliability by generating replicas on multiple racks. According to the replication policy, placement of 3 replicas is as follows:

- 1st replica on the local rack
- 2nd replica on the local rack of different machine
- 3rd replica on the different rack

But this policy may change in future.

3.1.6 HDFS File Read and Write [22]

In the Hadoop Cluster, Namenode accepts the request but does not directly read or write data to HDFS which is one of the reasons for HDFS‟s scalability. Initially, client interacts with the Namenode to update the HDFS namespace of Namenode and client retrieves block locations for reading and writing then it directly interacts with Datanode to read/write data. The Read and Write operations on the file are explained below.

3.1.6.1 HDFS Write

The write operation in HDFS is done in seven steps as shown in figure 9.

1. Create new file in the Namenode‟s Namespace and calculate block topology
2. Stream data to the first datanode
3. Stream data to the second datanode in the pipeline
4. Stream data to the third datanode
5. Success/Failure acknowledgement
6. Success/Failure acknowledgement
7. Success/Failure acknowledgement

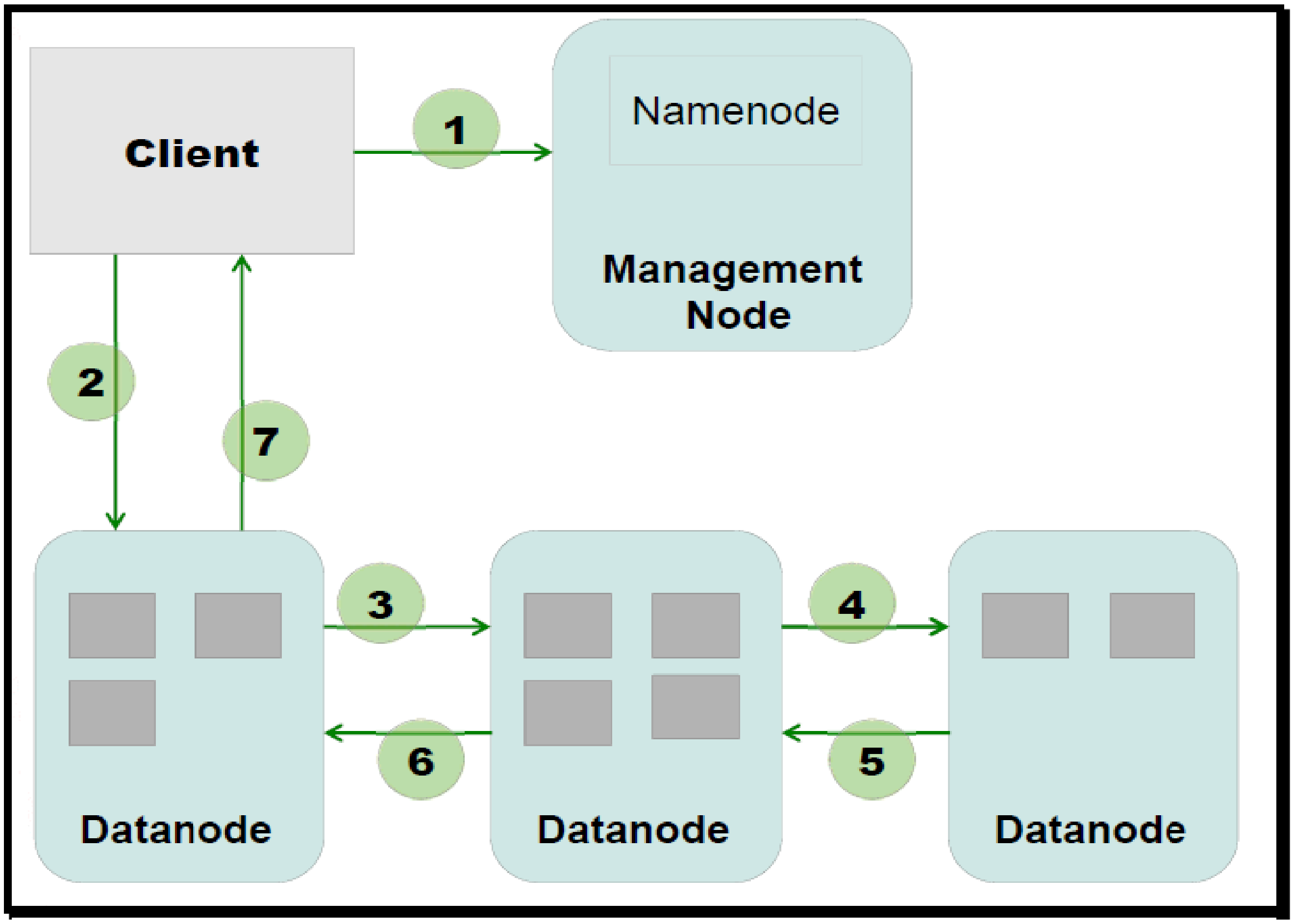

Figure 4 HDFS Write [21]

3.1.6.2 HDFS Read

The read operation in HDFS is done in three steps as shown in figure 10.

1. Client retrieves block location from Namenode
2. Client read blocks to re-assemble the file
3. Client read blocks to re-assemble the file

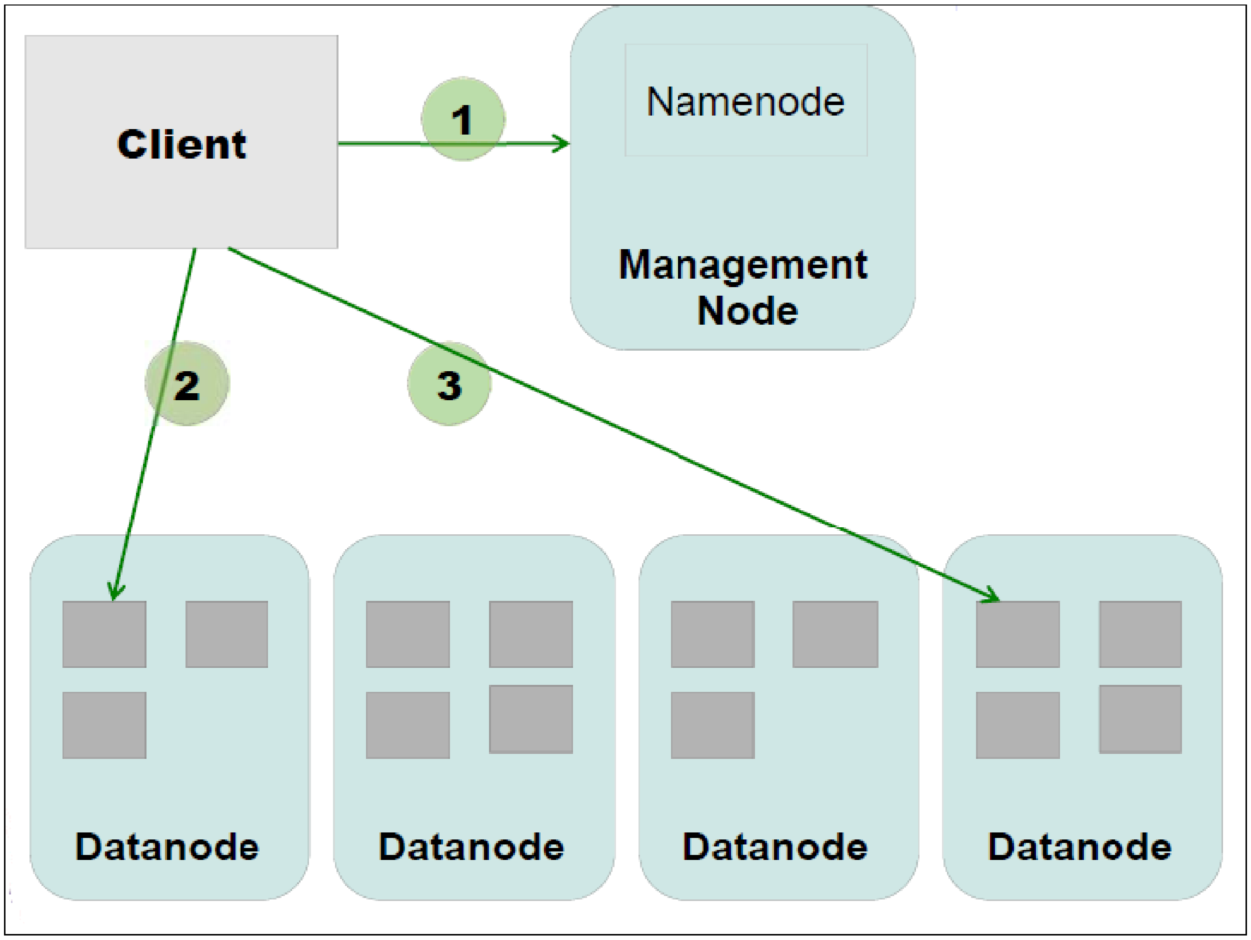

Figure 5 HDFS Read [21]

3.1.7 Difference between GFS and HDFS

Hadoop Distributed File System is based on Google File System but there are few differences that can be noticed between both of them. In HDFS, only single-writers per file is permitted whereas GFS supports multiple-writers. HDFS does not has any append operation but new versions supports appending at the end of file and not at any offset whereas GFS has append operation which can append at any offset.

HDFS also has few advantages over GFS. HDFS is Open source and provides many interfaces (like Thrift (C++, Python), libhdfs (C), FUSE) and libraries for different file systems like KFS, S3, etc. which is not the case with GFS i.e. GFS is not Open source and does not provide various interfaces or libraries for different file systems.

3.1.8. Disadvantages of HDFS

HDFS has a disadvantage that its Namenode daemon process must be running throughout working of cluster because if this process crashes then cluster is down. Namenode is a single point of failure in Hadoop Cluster so it should be on a reliable hardware which can sustain a disk failure. Usually Namenode failure is not an issue. But Hadoop version above 2 provides high availability of Namenode through an active standby which is always running and takes over if Namenode fails. The version used in our setup is 2.7.2 which is supported by active standby .

3.1.9 MapReduce

It is a distributed data processing framework which makes it easy to implement distributed applications that runs on a cluster. It processes large amount of data in-parallel on large clusters of commodity hardware in a fault-tolerant and reliable manner.

MapReduce reflects an associated implementation for generating and processing large data-sets and a programming model. User specified map functions processes a [key, value] pair and generates a set of intermediate [key, value] pairs while user specified reduce functions merge all intermediate values corresponding to the same intermediate key. Many (but not all) real-world application tasks fit this programming model and hence can be executed in a Hadoop MapReduce environment.

A MapReduce job usually splits the input data-set into independent chunks of data which are processed by the map tasks in parallel as shown in figure 6. The outputs of the maps are then sorted by the framework, which are then given as input to the reduce tasks. Generally, both the input and the output of the job are stored in a file-system. The framework takes care of monitoring tasks,scheduling them and re-executes the failed task [19].

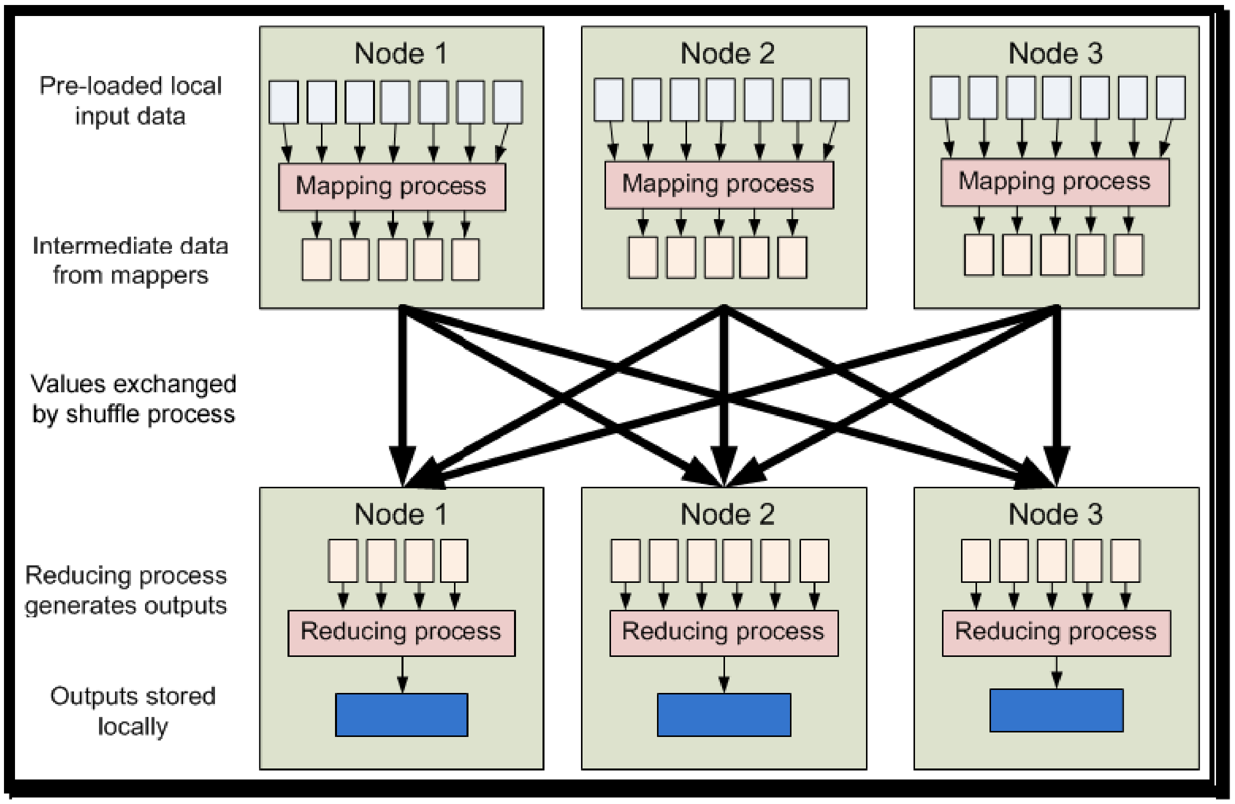
Pre-loaded local input data
Intermediate data from mappers
Values exchanged by shuffle process
Reducing process generates outputs
Outputs stored locally
Node 1
Node 2
Node 3
Mapping process
Reducing process

Figure 6 MapReduce [23]

In Hadoop, the MapReduce engine is implemented by two software services, the JobTracker and TaskTracker. The centralized JobTracker runs on a dedicated cluster node and is responsible for splitting the input data into pieces for processing by independent map and reduce tasks (by coordinating with the user-level file system),scheduling each task on a cluster node for execution, monitoring execution progress by receiving heartbeat signals from cluster nodes, and recovering from failures by re-running tasks. On each cluster node, an instance of the TaskTracker service accepts map and reduce tasks from the JobTracker. By default when a new task is received, a new JVM instance will be spawned to execute it. Each Task-Tracker will periodically contact the JobTracker via a heartbeat message to report task completion progress and request additional tasks when idle [23].

3.1.9.1. Map Task

The Map task can be decomposed into 5 phases which are explained below [23]and shown in figure 7 .

1. A read phase where the input split is loaded from HDFS (Hadoop Distributed File System) and the input key-value pairs (records) are generated.
2. A map phase where the user-defined and user-developed map function is processed to generate the map-output data.
3. A collect phase, focusing on partitioning and collecting the intermediate (map output) data into a buffer prior to the spilling phase.
4. A spill phase where (if specified) sorting via a combine function and/or data compression may occur. In this phase, the data is moved into the local disk subsystem (the spill files).
5. A merge phase where the file spills are consolidated into a single map output file. The merging process may have to be performed in multiple iterations.

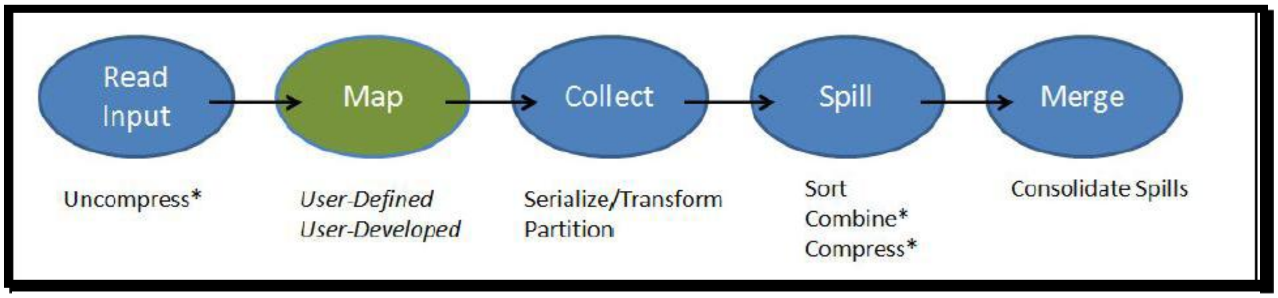

Figure 7 Map Task Execution [24]

3.1.9.2 Reduce Task

The Reduce task can be framed in 4 phases as given below [9][19] and shown in figure 8.

1 1. A shuffle phase where the intermediate data from the mapper nodes is transferred to the reducer nodes. In this phase, decompressing the data and/or partial merging may occur as well.

2. A merge phase where the sorted fragments (memory/disk) from the various mapper tasks are combined to produce the actual input into the reduce function.

3. A reduce phase where the user-defined and user-developed reduce function is invoked to generate the final output data.

4. A write phase where data compression may occur. In this phase, the final output is moved into HDFS.

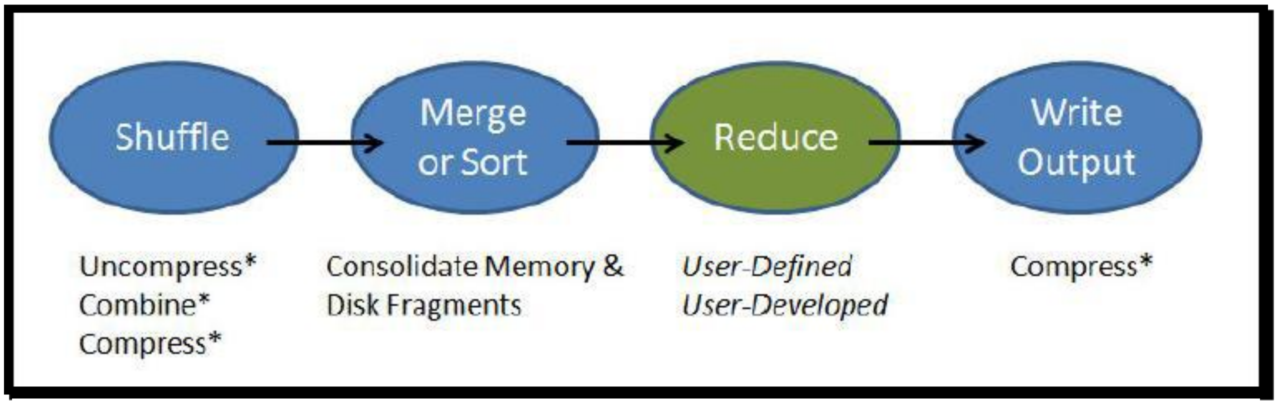
Shuffle
Merge or Sort
Reduce
Write Output
Uncompress* Combine* Compress*
Consolidate Memory & Disk Fragments
User-Defined User-Developed
Compress*

Figure 8 Reduce Task Execution [24]

In our project, MapReduce feature is being used for parallel processing of records for situation i.e each data record is mapped to some star(having centroids) and minimum distance is calculated, and is attached to the key of the star the reducer then clubs all the distances associated with a particular key and then repeats this for all stars and returns the inter-cluster distance. MapReduce will be used in both updating the position of stars and calculating fitness and returns result in minimum time.

3.1.10 Inefficiencies in Hadoop

Three important inefficiencies in Hadoop‟s design which can be noticed are :

- Delayed speculative execution: one speculative execution decision severely delaying or even precluding subsequent speculative executions at great overall costs for the job running time
- The lack of sharing failure information is that multiple tasks could be left wasting time re-discovering a failure that has already been identified by another task
- Induced reducer death problem: from only the news of a connection failure Hadoop cannot reliably distinguish an underlying cause. We show that this limitation unnecessarily introduces additional failures into the system. Specifically, otherwise localized failures involving a compute node can propagate to tasks running on healthy nodes.

These inefficiencies are recovered in the upcoming versions of Hadoop and are handled in following manner:

- The JobTracker runs a speculative execution algorithm which attempts to improve job running time by duplicating underperforming tasks.
- Failure of tasktracker is detected by polling after every 200s and if no response is received till 600s then that tasktracker is declared dead and its tasks are restarted on another node.
- Failure of data node is detected by time out and connection errors.

3.2 DATA CLUSTERING

The process of grouping a set of physical or abstract objects into classes of similar objects is called clustering. It is defined as organizing the data into groups so that the data in one group have higher similarity index while the members of different groups have higher dissimilarities. Now the similarity is estimated on the basis attribute values of the objects. A cluster of objects is collectively treated as a single group in many applications. As compared to classification the label of the class is unknown in cluster analysis. [25]Clustering has its roots in many areas, some typical examples could be, In medical areas to categorize the genes with similar attributes and to discover knowledge about the structures that are inherent in population also say to detect tumor in a human brain the RGB values are compared to the a healthy brain after performing data clustering.

The business analysts have to characterize customer groups on the basis of their purchase patterns, clustering can help them to discover the distinct groups of customer. It helps in classifying the web documents for the discovery of information.

3.2.1 TYPES OF CLUSTERING

There are a number of data clustering algorithms available and the choice of the algorithm depends upon the purpose of clustering, type of data available and the application. According to In general major clustering algorithms are broadly defined as under-

Partitioning Methods: In a database of n data objects , partitioning method constructs K groups of the data, where each group depicts a cluster. And K<=n. It should satisfy the following requirements:

- Every group must contain atleast on data object.
- Each object must belong to only one group.

The partitioning method [26] creates an initial partitioning, given k number of partitions to be constructed.After that it uses a iterative relocation technique to improve the partitioning by moving tghe objects from one cluster to another cluster. General criteria of a good partition is that the objects in a group are more close whereas the objects of different cluster are far apart and are very different. We can judge the quality of partitions by various criterions.

Most applications adopt popular heuristic methods like in the k-means algorithm, each cluster is represented by the mean value of the objects in the cluster.

In the k-mediods algorithm each cluster is depicted by the one of the objects located near the center of the center. These type of heuristic data clustering methods work well for spherical shaped clusters for small to medium sized datasets.

Hierarchical Methods: These type of method creates a hierarchical decomposition of the given dataset objects. It can be classified into agglomerative or divisive based on how decomposition is to be done.In the agglomerative approach, bottom up strategy is followed i.e. starting from a data object and forming a separate group. It then untermittedly merges the objects of the group close to each other until all of the groups merges into one , till the termination condition is met or the one in the topmost hierarchy is reached. The divisive approach is also known as the top-down approach, the iteration starts with all the objects in one cluster and in successive iteration a cluster is split into smaller clusters until each object is a separate cluster or the termination condition is met. Hierarchical methods experience a problem from the fact that once a merge or a split is done it can never be undone i.e. they cannot correct erroneous decisions.

Density Based Methods: The conventional partitioning methods only spherical-shaped cluster as they are distance based, these methods encounter difficulty in discovering clusters of arbitrary shapes. There are clustering methods which have been developed based on the notion of density. [25] Their basic idea is to continue growing a given cluster as long as the density (number of objects or data points) in the “neighborhood” exceeds some threshold. For example, for each data point within a given cluster, the neighborhood of a given radius has to contain at least a minimum number of points. Such a method can be used to filter out noise or outliers and discover clusters of arbitrary shape. Density-based methods can divide a set of objects into multiple exclusive clusters, or a hierarchy of clusters. Typically, density-based methods consider exclusive clusters only, and do not consider fuzzy clusters. For example DBSCAN is a density based method that grows clusters according to the density

Grid Based Methods: This type of clustering uses a multi-resolution grid data structure. . The main advantage of this method is its fast processing time, which is typically independent of the number of data objects and dependent only on the number of cells in each dimension in the quantized space. Using grids is often an efficient approach to many spatial data mining problems, including clustering. Grid-based methods can be integrated with other clustering methods such as density-based methods and hierarchical methods.

Model Based Methods: These type of methods hypothesize a model for each of the cluster and find the best fit of the data to the given model. It locates clusters by constructing a density function that reflects the spatial distribution of the data points. It also leads to a way of automatically determining the number of clusters based on standard statistics, taking “noise” or outliers into account and thus yielding robust clustering methods.

As mentioned in the literature survey meta-heuristic algorithms prove to be very efficient in terms of speedup in the convergence of the algorithms. Hence we have taken the black hole algorithm [27] as an inspiration for data clustering. Following is a short description of the phenomenon and the algorithm.

3.2.2 BLACK HOLE ALGORITHM:

Black hole phenomenon:

A black hole in space comes into existence when a massive star collapses. The gravitational pull of a black hole is too high. Any body that crosses the boundary of black hole is gulped by the black hole and the body vanishes with the speed of light.

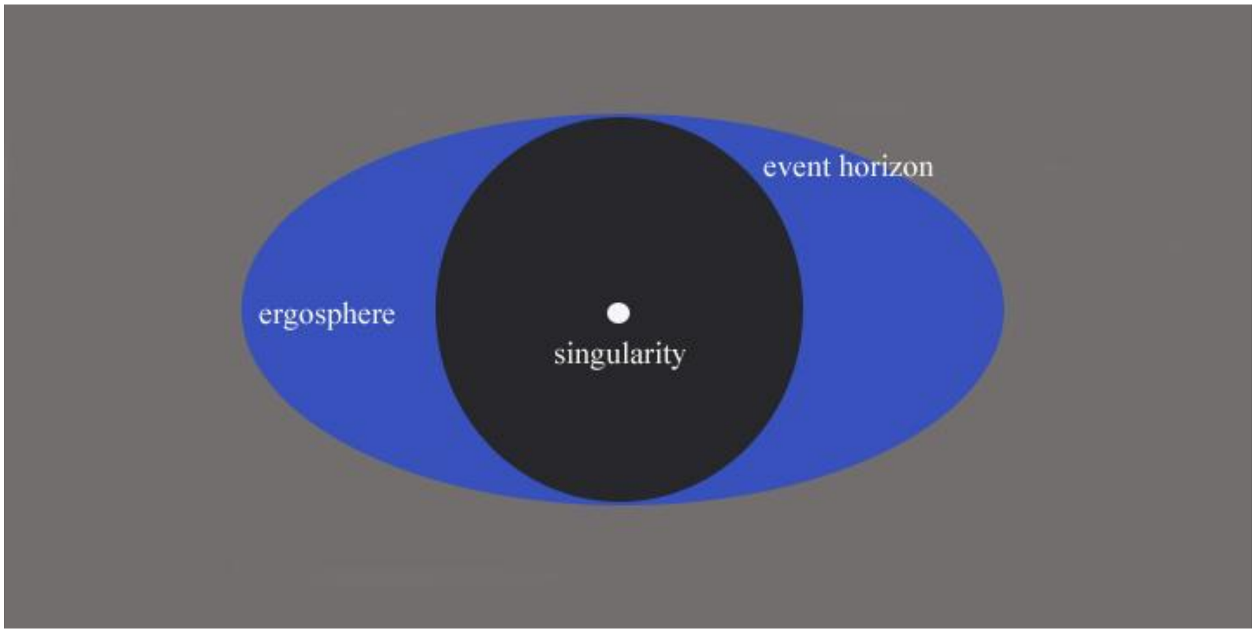

Figure 9 Black Hole with its event horizon

At the centre of the black hole lies the singularity region. This region contains all the mass of the black hole and it can be thought as of having infinite density. The rotating black hole is surrounded by a region known as ergosphere. It is a volume whose inner boundary is the black hole's event horizon. The boundary of the black-hole is spherical-shaped and is known as event horizon. It is coined as schwarzchild radius by the physicists and is calculated by the equation:

$$R = \frac{2GM}{c^2} \qquad -(1)$$

Where G is the gravitational constant. M is the mass of black hole and c is the speed of light.

Black hole algorithm: The black hole algorithm is population based algorithm. In this the population of candidate solutions known as stars, is generated and is distributed randomly in the search space. [27]After the initialization of population is done, the fitness values are evaluated. The candidate with the best or minimum fitness value is qualified as black hole while the others are normal stars.The black hole now starts absorbing stars near to it. All the stars move towards the black hole according to the equation:

$$p_i(t+1) = p_i(t) + \left(p_{BH} - p_i(t)\right) * rand \quad i = 1,2 \ldots N \qquad - (2)$$

Where $p_i(t+1)$ and $p_i(t)$ are the locations of the ith star at the iterations t and $t+1$. P_{BH} is the location of black hole in the universe and rand is random number between 0 and 1, N is the number of stars.

While moving the stars it may happen that the star crosses the event horizon of the black hole, then the black hole swallows the star and the star dies, at the same time a new star is created in the search space. This is done so that number of stars remain constant.

The radius of the event horizon of the black hole is calculated by the following equation:

$$r = \frac{fit_{BH}}{\sum_{i=1}^{N} fit_i} \qquad - (3)$$

Where fit_{BH} is the fitness value of black hole and fit_i is the fitness value of stars and N is the number of stars.The basic flow of the algorithm is depicted as under-

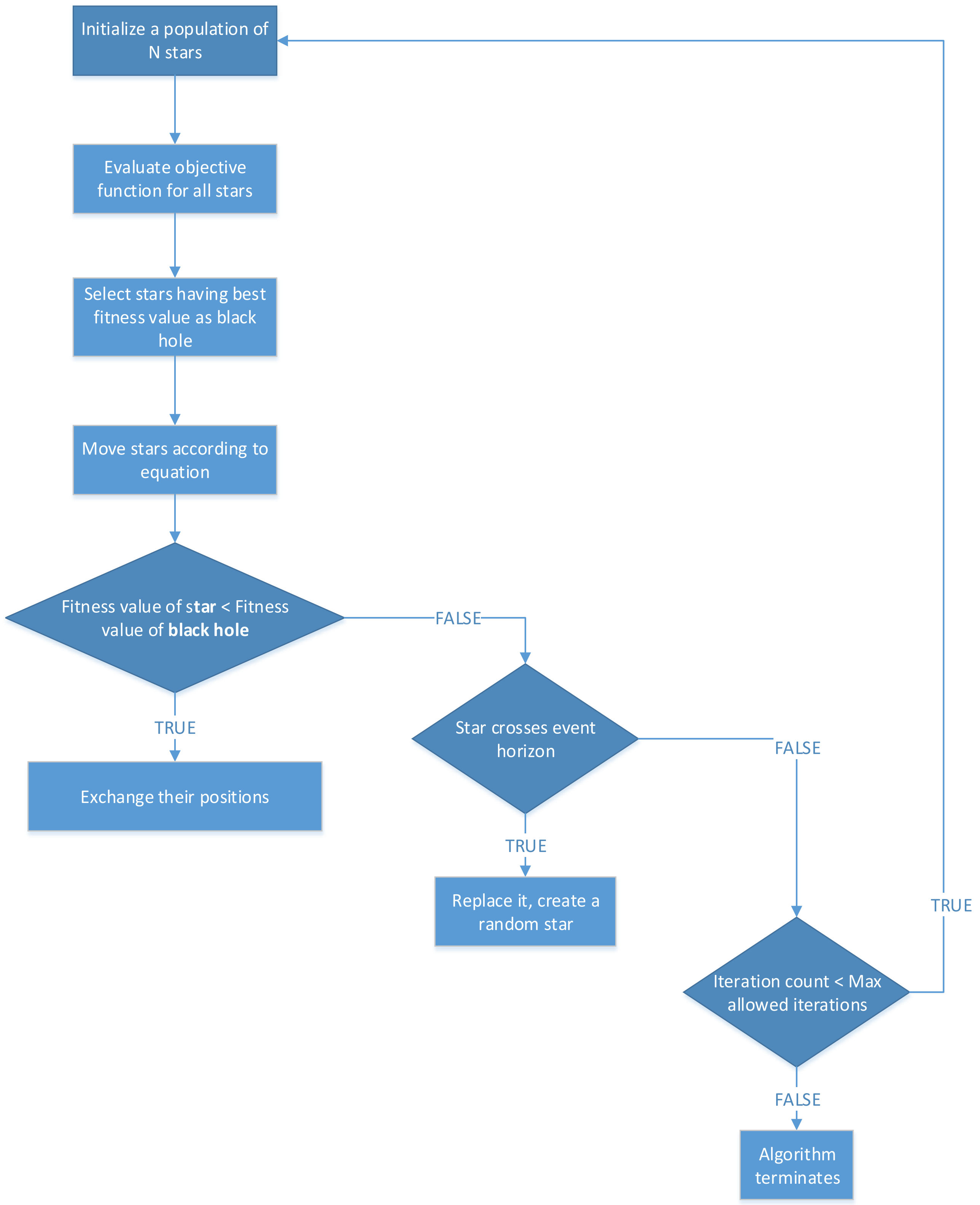
Initialize a population of N stars
Evaluate objective function for all stars
Select stars having best fitness value as black hole
Move stars according to equation
Fitness value of star < Fitness value of black hole
FALSE
TRUE
Exchange their positions
Star crosses event horizon
FALSE
TRUE
Replace it, create a random star
TRUE
Iteration count < Max allowed iterations
FALSE
Algorithm terminates

Figure 10 Flowchart of Black Hole algorithm

3.3 Test Datasets-

The proposed algorithm is tested on five benchmark datasets acquired from UCI machine learning repository.

SNo	Datasets	No: of clusters	No: of features	No: of data objects
1	Iris	3	4	150
2.	Glass	6	9	214
3.	Wine	3	13	178
4.	Magic	2	10	19,020
5.	Poker Hand	10	10	1,000,000

Table-4.1 Characteristics of test datasets.

1.Iris Dataset: It is flower dataset having four attributes as petal length and width and sepal length and width. The dataset contains three classes of 50 instances each.

2. Glass Dataset: The glass dataset by USA Forensic Science Service contains 6 types of glass in terms of oxide content. There are in all ten attributes but we are using nine attributes for clustering as we are using numerical data only for clustering.

3. Wine Dataset: It is a dataset containing the data of analysis of chemical determining the origin of wines The analysis gives the quantities of 13 constituents found in three types of wines.

4. Magic Dataset: It is a Magic gamma telescope dataset generated to simulate registration of high energy gamma particles in an atmospheric telescope. Again we are using only 10 attributes from 11 , as we only need numerical data for clustering.

5. Poker hand Dataset: In this dataset each record is an example of a hand consisting of five playing cards drawn from a standard deck of 52. Each card is described using two attributes suit and rank, for a total of 10 predictive attributes.

CHAPTER 4

PROPOSED WORK

In the previous chapter, description of the sequential black hole algorithm was given, which is one of the meta-heuristic algorithm. Now the problem with meta-heuristic algorithms is that they are very computation intensive. Also major clustering algorithms are designed for centralized system. If the memory and CPU capacity is less for the input dataset then the clustering will be a laborious task. Now to leverage the strength of the sequential black hole algorithm it is implemented on MapReduce architecture of hadoop to accelerate the speed of clustering. The parallel black hole MapReduce algorithm inherits the characteristics of the black hole algorithm, as no parameters are to be set manually. To evaluate the performance of the proposed algorithm, several datasets are usedwith different numbers of nodes. Experimental results show that the proposed algorithm can provide a significant speedup as the number of nodes increases.
The alpha black hole MapReduce algorithm : As the major drawback of the black hole algorithm was that it took a lot of time in computing the fitness values for the stars, which detroited the overall performance of the algorithm. Hence the task of computing the fitness values for the entire population of stars is accomplished using the powerful MapReduce architecture of the hadoop framework and optimization of objective function is achieved. While the population updation and merging the results is done in the drivercode as the population will be always less in size as compared to the size of the input dataset.

The reason to implement the and Fitness computation module on MapReduce is to improve the ability of the black hole algorithm on mining large-scale dataset.For the proposed algorithm, the key value pair of MapReduce is associated with a star identified by a numerical ID named starID as the key and the star information as the value of the pair. The star information is a set of values about star ID (starID), current star location (star loc), current star fitness value (star fit). An extra alpha parameter is introduced to improve the exploitation and exploration of the algorithm and can be a set to a constant value.

$$\alpha = \alpha - curr\_iter * (\alpha/\max\ \_iter) \qquad -(4)$$

$$w = \alpha * rand\_num \qquad -(5)$$

And the equation of movement is modified as under:

$$p_i(t+1) = p_i(t) + \left(p_{BH} - p_i(t)\right) * w \quad i = 1,2 \ldots N \qquad -(6)$$

The star location is the structure of the cluster centroids. The stars and the input dataset are the input to the Fitness computation module, which is responsible for calculating the fitness of each star using the objective function of square of Euclidean distance between star and the data point using Eq-7.

$$F(C,D) = \sum_{i=1}^{N}\sum_{j=1}^{K}(C_i - D_j)^2 \qquad -(7)$$

After Fitness computation module is finished with its work, the Merge function refreshes the information of stars by combining the updated population of stars and the computed fitness values by the reducer and then sends the stars to the next iteration. A check of event horizon id also applied before the start of next iteration using equation-(3).

4.1 Pseudocode of the algorithm is depicted as under-

Pseudo code for the α-black hole MapReduce algorithm:

```
1. Initialize the population.
2. /*Map Function */
   2.1 Map(Key: dataID, Value: data)
   2.2 dataid=key
   2.3 data_val=value
   2.4  //get the population of stars
   2.5  //calculating minimum distance
   2.6  for each  star
   2.7       mindistance=getmindistance (data,star.star_loc)
   2.8       emit(starid,mindistance)
   2.9 endfor
3. /* Reduce Function*/
   3.1 Reduce(Key:starid,  Value:mindistance_list)
   3.2 For each mindistance in the list mindistance_list
   3.3      Sum+= mindistance
   3.4 endfor
   3.5 emit(starid,sum)
4. Select the star with minimum fitness value as black_hole
5. Update population of stars using equation –(6)
6. Check for event horizon
   1.1 distance=dist(star.star_loc, black_hole.loc
   1.2 radius=event_horizon(star.star_fit,black_hole.fit)
   1.3 if(distance <radius)
   1.4     generate a new star at a new location randomly
   1.5 endif
   1.6 star.update(starid)
6. Repeat until max iteration is reached or the desired result is achieved.
```

4.2 Flowchart of the α-Black Hole Algorithm

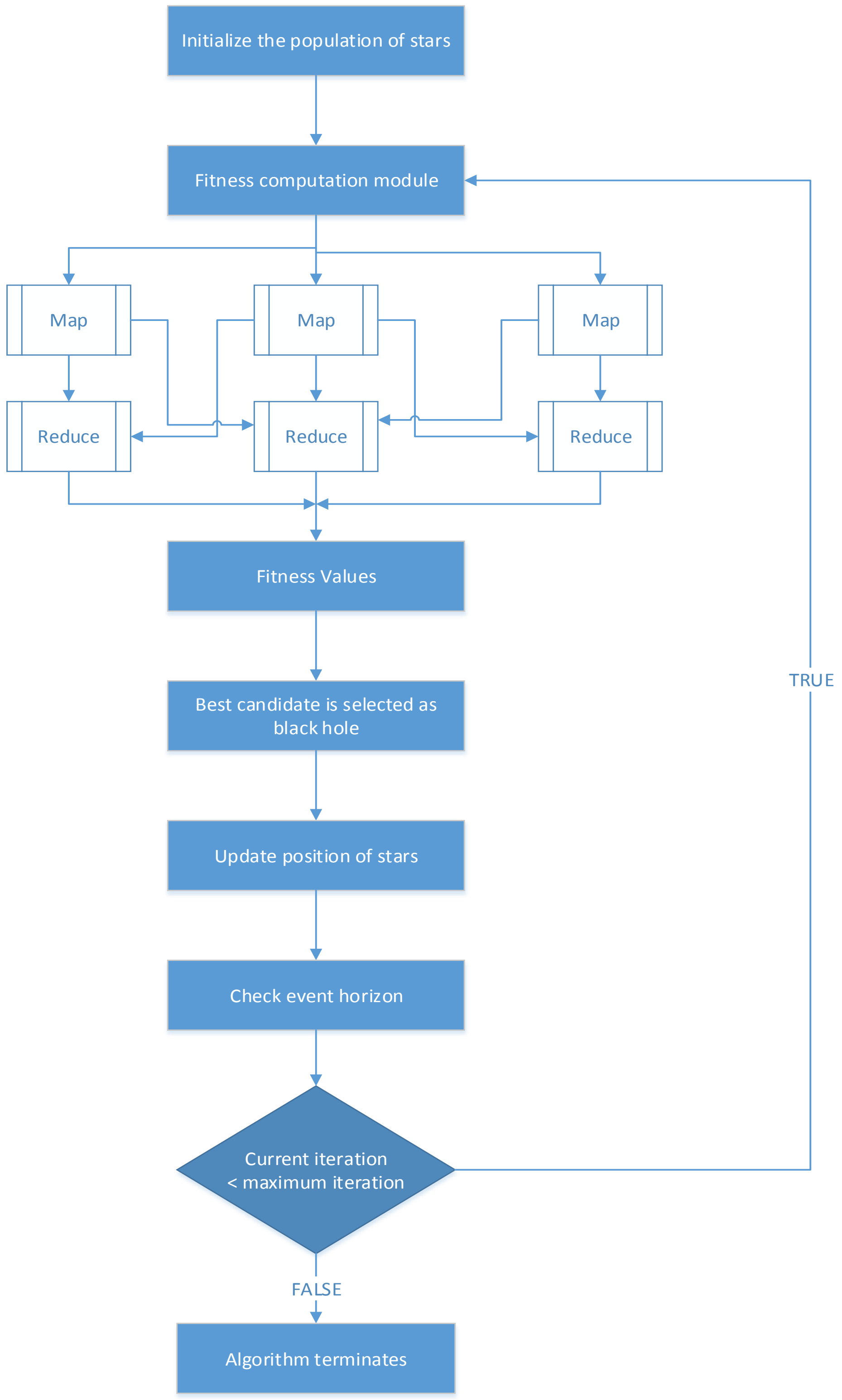
Initialize the population of stars
Fitness computation module
Map
Map
Map
Reduce
Reduce
Reduce
Fitness Values
Best candidate is selected as black hole
Update position of stars
Check event horizon
Current iteration < maximum iteration
TRUE
FALSE
Algorithm terminates

Figure 11 Flowchart for the α-black hole algorithm

CHAPTER 5

EXPERIMENTAL RESULTS

5.1 SIMULATION SETUP

The experimental setup is a multimode hadoop cluster of three nodes. Apart from that a single node cluster is also used for comparison purpose. The operating system used is ubuntu 14.10. The configuration of the systems is given in the table -

SNo	**Name**	**RAM**	**CPU**	**HDD**
1.	Single Node Cluster 64-bit System	4.00 GB	Intel CORE i3 @ 2.40 GHz	300 GB
2.	Master Node	4.00 GB	Intel CORE i5 @ 3.20 GHz	500 GB
3.	Slave 1	4.00 GB	Intel CORE i5 @ 3.20 GHz	500 GB
4	Slave 2	4.00 GB	Intel CORE i5 @ 3.20 GHz	500 GB

Table -5.1 Configuration of systems used.

5.2 Performance Evaluation

The performance of the α-black hole MapReduce algorithm can be evaluated by calculating the sum of intracluster distance- which is the distance between each data object and the center of the corresponding cluster is calculated and summed up using Eq-7. Clearly, the smaller the sum of intra-cluster distances, the higher the quality of the clustering. The sum of intra-cluster distances is also the evaluation fitness in this work.

The sum of intra-cluster distances obtained by algorithms on different datasets is depicted as under:

SNo	Criteria	Datasets	k-means	Black hole Algorithm	α-black hole MapReduce algorithm
1.	Best	Iris	105.72902	101.9571	96.40594
	Average		128.40420	117.0912	97.07521
2.	Best	Glass	227.97785	215.6773	211.6458
	Average		260.83849	230.4978	214.8541
3.	Best	Wine	16,555.04499	16,601.4159	16,302.3696
	Average		16,963.67942	16,667.6325	16,332.8639
4.	Best	Magic	1,652,401.6848	1,523,053.4569	1,160,669.0871
	Average		1,842,508.5412	1,618,436.3174	1,291,461.512
5.	Best	Poker Hand	6,707,346.38	6,432,189.11	6,331,167.94
	Average		6,721,218.87	6,532,781.27	6,329,489.57

Table 5.2 sum of intra-cluster distances for various datasets

No: of Nodes verses of time plot for different benchmark datasets:

1.Iris dataset:

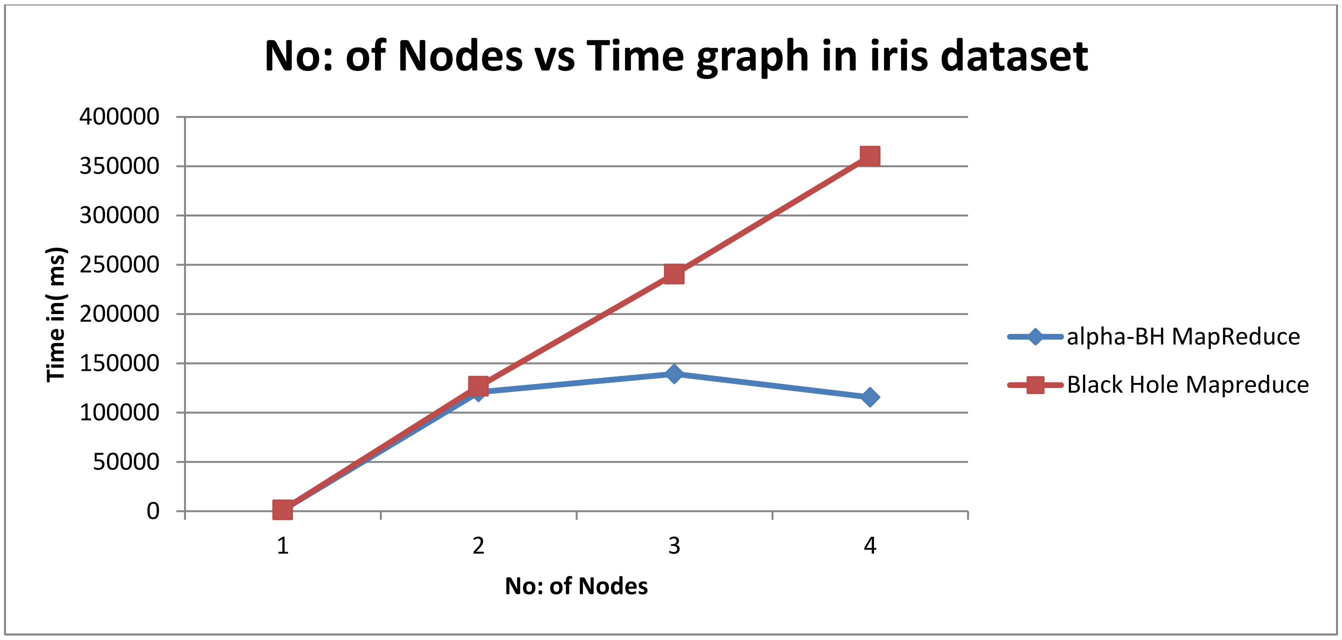

Figure 12 Nodes Vs time plot for iris dataset

2. Glass Dataset:

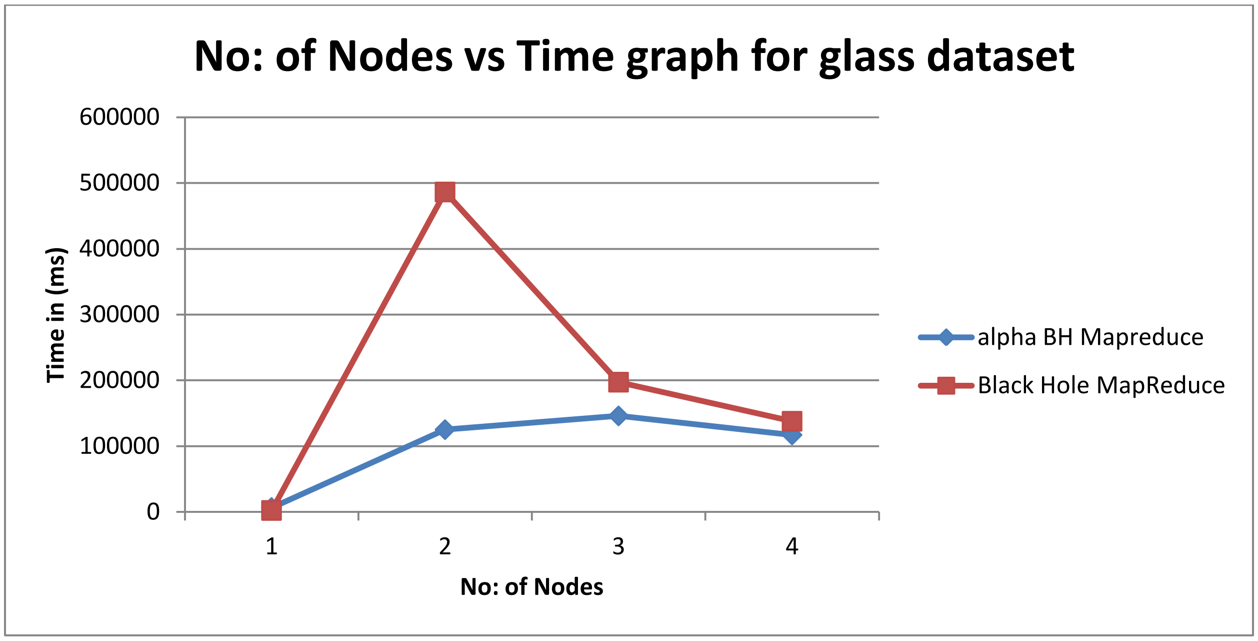

Figure 13 Nodes Vs time plot for glass dataset

3. Wine Dataset:

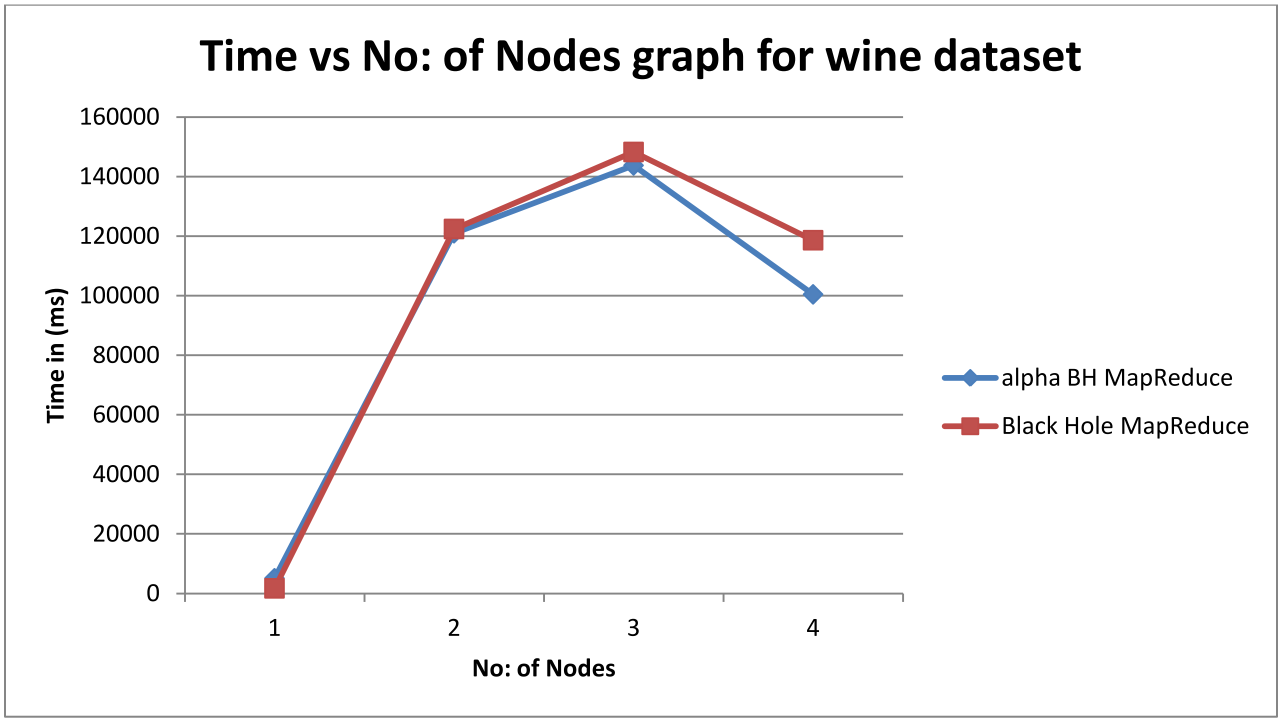

Figure 14 Nodes Vs time plot for glass dataset

4.Magic Dataset:

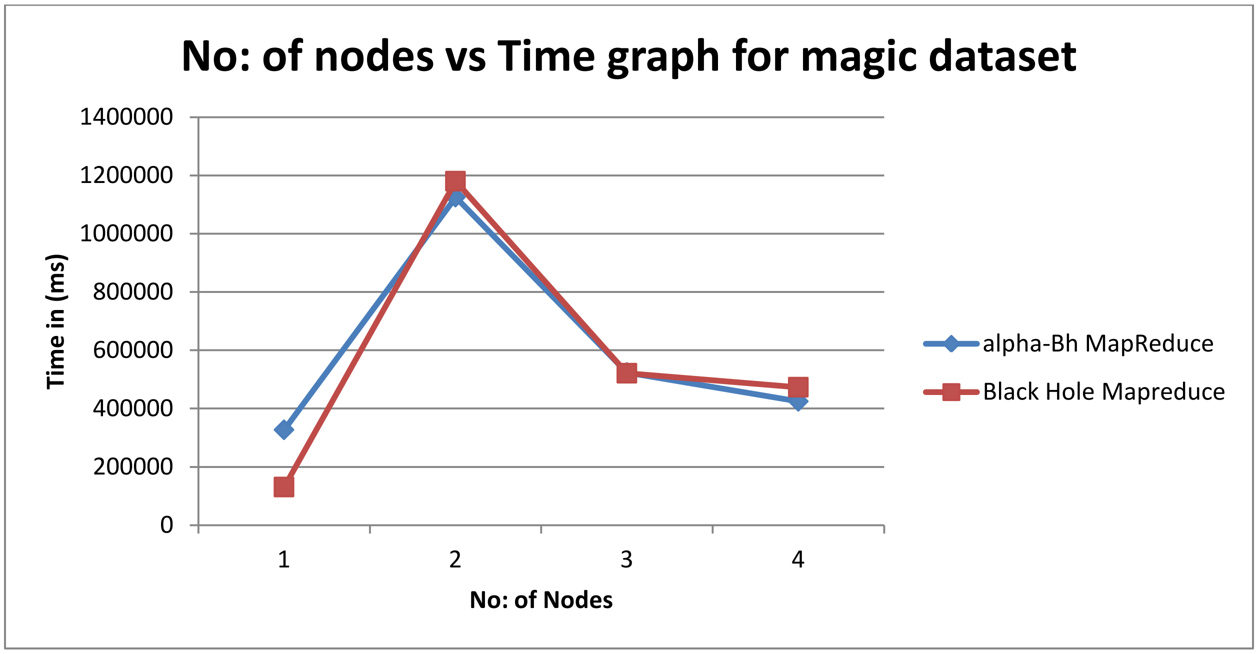

Figure 15 Nodes Vs time plot for magic dataset

5.Poker Hand Dataset:

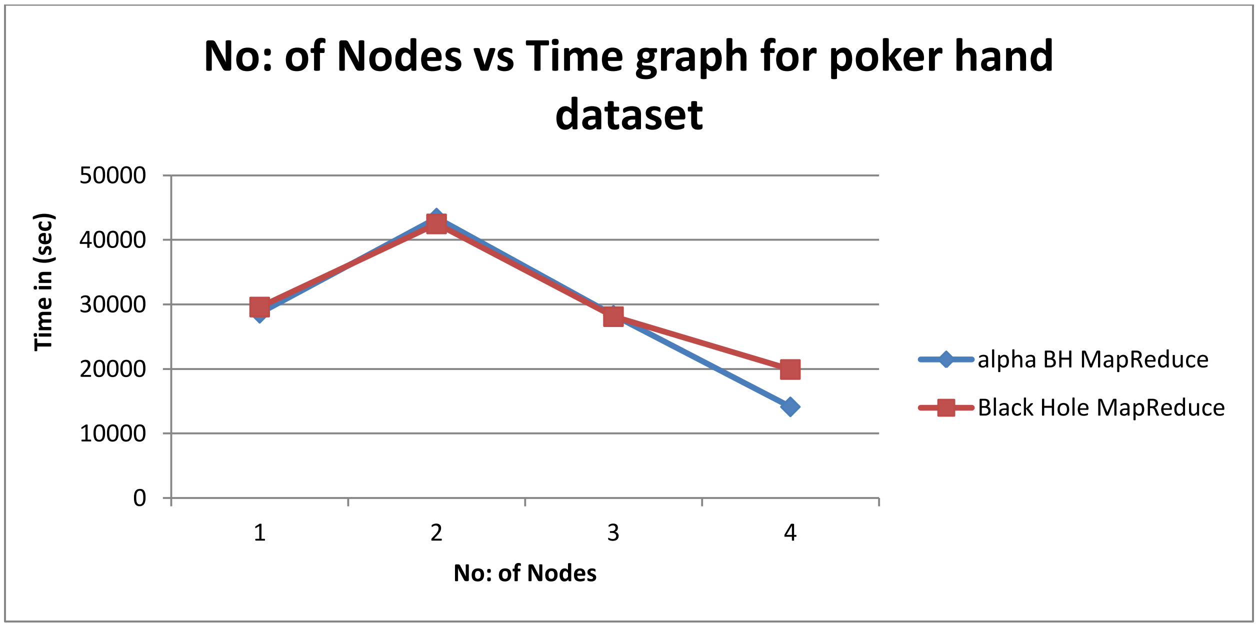

Figure 16 Nodes Vs time plot for poker hand dataset

5.3 Performance Evaluation for convergence- The convergence graph refers to the plot of number of iteration verses the fitness values obtained by various benchmark datasets.

1.Iris Dataset:

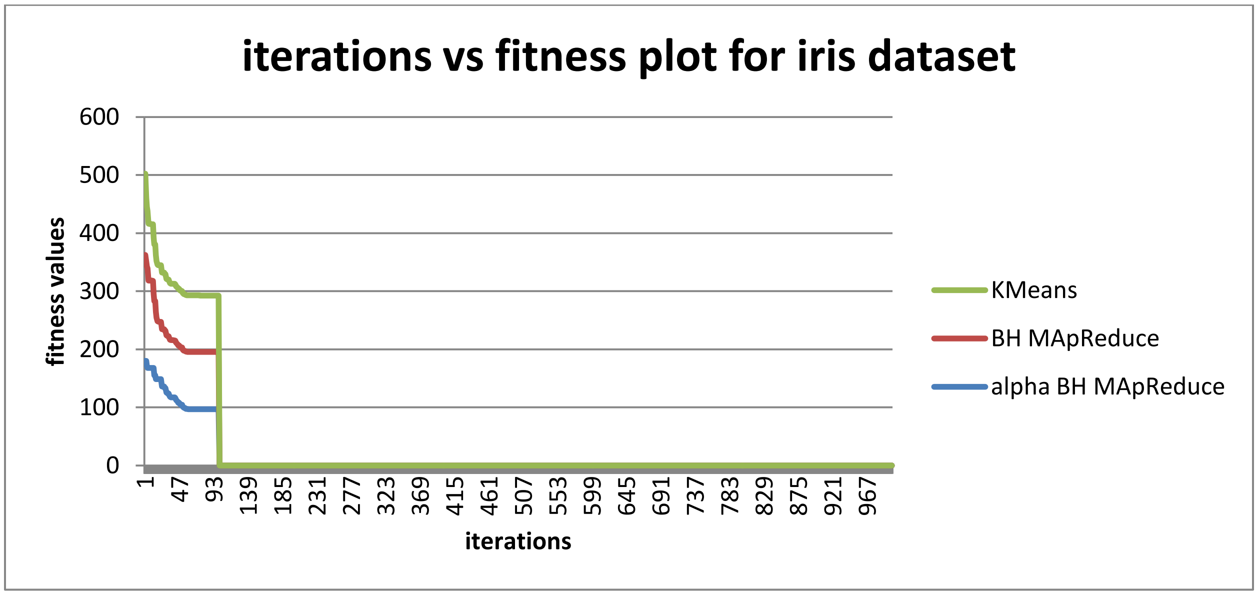

Figure 17 Iteration Vs Fitness plot for iris dataset

As we can see from the graph the algorithm converge early , around 100 iterations and after that it becomes constant. Also since the dataset is very small the performance is more or less as that of kmeans algorithm.

2. Glass Dataset:

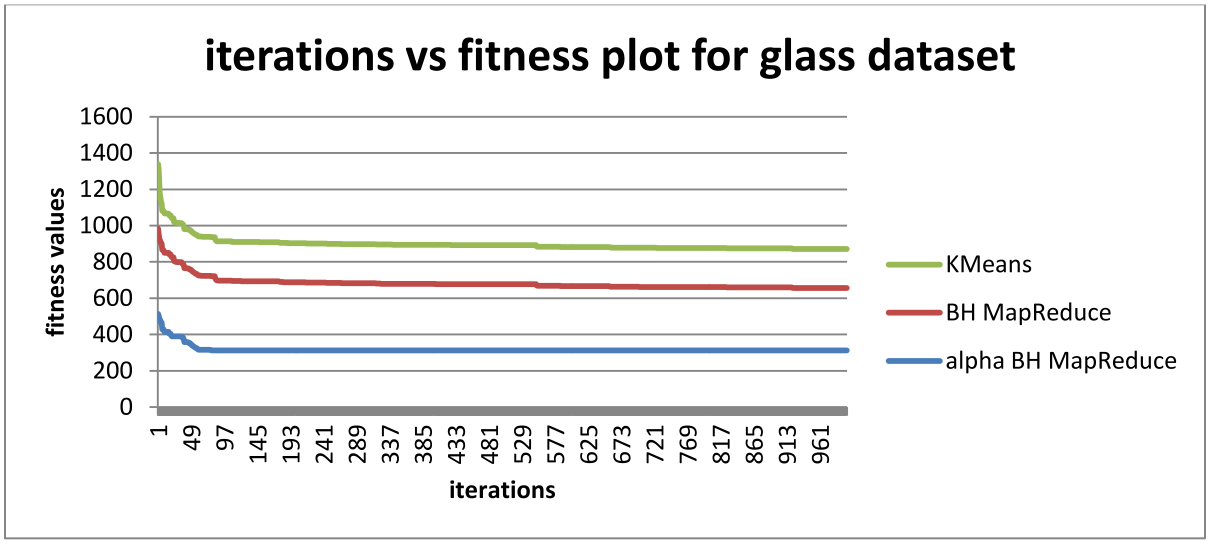

Figure 18 Iteration Vs Fitness plot for glass dataset

It can be clearly be seen from the graph that α-BH MapReduce outperforms the other two algorithms and becomes constant in further iterations.

3. Wine dataset:

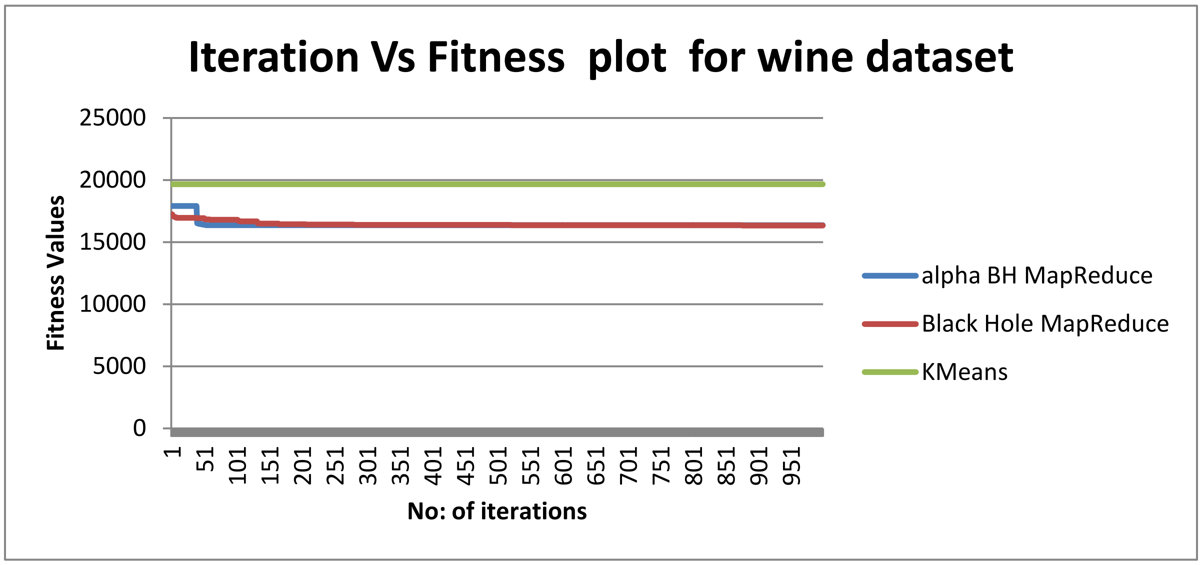

Figure 19 Iteration Vs Fitness plot for wine dataset

As shown in graph both the variants of black hole MapReduce algorithm outperform the kmeans algorithm. But there is very little difference in fitness values of between Black Hole MapReduce and α-BH MapReduce algorithm.

4.MagicDataset

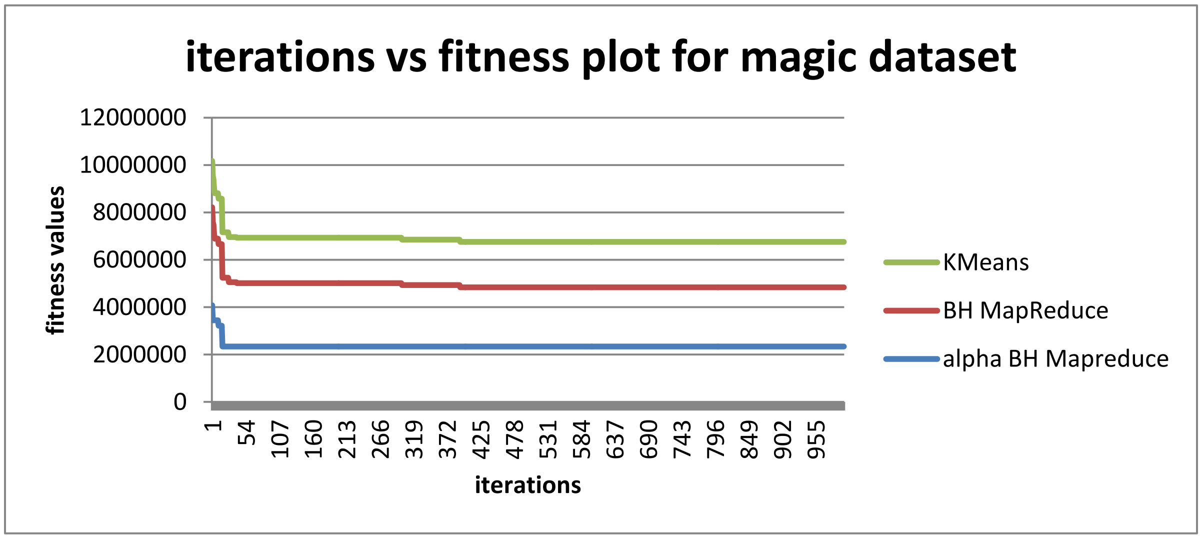

Figure 20 Iterations Vs Fitness plot for magic dataset

CHAPTER 6

CONCLUSION & FUTURE WORK

We dedicated our work in parallelization of black hole algorithm for clustering large data sets on Apache MapReduce architecture. It puts to advantage, distributed characteristics of MapReduce to parallelize black hole algorithm and the initial parameter free nature of black hole algorithm.
Derived results also support the idea that black hole is parameter free and is easy to implement and produces better results than conventional KMeans algorithm, KPSO etc. on five benchmark datasets.
The α-BH MapReduce algorithm was a variant which exploited the random nature of algorithm for better convergence and improving the exploitation and exploration of the algorithm. The future work will be to reduce the no: of iterations to better the computation time.

REFERENCES

[1] B. A. Wooley, "Scaling Clustering for the Data Mining," Department of Computer Science, Mississippi State University, [Online]. Available: http://www.cs.utexas.edu/users/csed/doc_consortium/DC99/wooley-abstract.html.

[2] B. Walker, "BIGDATA STATISTICS," vouchercloud, [Online]. Available: http://www.vcloudnews.com/every-day-big-data-statistics-2-5-quintillion-bytes-of-data-created-daily.

[3] Wikipedia, "MapReduce," [Online]. Available: https://en.wikipedia.org/wiki/MapReduce.

[4] R. A. Formato, "Central force optimization: A new metaheuristic with applications in applied electromagnetics," in *ResearchGate*, 2007.

[5] B. B. D. Karaboga, "On the performance of artificial bee colony (ABC) algorithm," *Elsevier,* p. 687–697, 2008.

[6] S. Mirjalili, "Moth-flame optimization algorithm: A novel nature-inspired heuristic," *Elsevier,* p. 228–249, 2015.

[7] S. S. U. M. Sanghamitra Bandyopadhyay, "A Simulated Annealing-Based Multiobjective objective Algorithm," in *IEEE*, 2008.

[8] S. Mirjalili, "The Ant Lion Optimizer," *Elsevier,* pp. 80-98, 2015.

[9] E. R. S. Saryazdi and H. N. pour, "GSA: A Gravitational Search Algorithm," in *Elsevier*, Iran, 2009.

[10] S. A. Mohammadia, M. Rahmani and M. Azadi, "Optimization of continuous ranked probability score using PSO," in *Growing Science*, Iran, 2015.

[11] X.-S. Yang, "A new meta-heuristic bat inspired algorithm," in *Springer*, 2010.

[12] R. Rajabioun, "Cuckoo Optimization Algorithm," *Elsevier,* p. 5508–5518, 2011.

[13] D. E. GOLDBERG and J. H. HOLLAND, "Genetic Algorithms and Machine Learning," in *IEEE*, Netherlands, 1988.

[14] X.-. S. Yang, "FireFly Algorithm, Flights and optimization," in *Springer*, 2010.

[15] I. E. Osman K. Erol, "A new optimization method: Big Bang–Big Crunch," in *Elsevier*, Turkey, 2006.

[16] J. Wang, D. Yuan and M. Jiang, "Parallel K-PSO Based on MapReduce," in *IEEE*, jinan, china, 2012.

[17] G. S. S. R. K. R.Bhavani, "A Novel Parallel Hybrid K-means-DE-ACO Clustering Approach for Genomic Clusterin using MapReduce," in *IEEE*, 2011.

[18] A. Hatamlou, S. Abdullah and H. N. pour, "A combinedapproachforclusteringbasedon K-means andgravitational," *Elsevier,* pp. 47-52, 2012.

[19] T. White, Hadoop, Defintive guide, O'reilly, 2012.

[20] R. ,. A. ,. F. Ivanilton Polato, "A comprehensiveviewofHadoopresearch—A systematic lietrature review," *Elsevier,* p. 1–25, 2014.

[21] "Hadoop Tutorial," [Online]. Available: http://www.coreservlets.com/hadoop-tutorial/.

[22] M. Y. Eltabakh, "Hadoop: A Framework for Data-Intensive Distributed Computing," 2012. [Online]. Available: http://web.cs.wpi.edu/~cs561/s12/Lectures/6/Hadoop.pdf.

[23] A. S. foundation, "MapReduce Tutorial," [Online]. Available: https://hadoop.apache.org/docs/current/hadoop-MapReduce-client/hadoop-MapReduce-client-core/MapReduceTutorial.html.

[24] "Google images," [Online]. Available: www.googleimages.com.

[25] J. han and M. Kamber, Data Mining Concepts and Techniques, San Francisco, USA: Elsevier, 2004.

[26] D. Sisodia and L. Singh, "Clustering Techniques: A Brief Survey of Different Clustering Algorithms," *International Journal of Latest Trends in Engineering and Technology (IJLTET),* no. ISSN: 2278-621X, pp. 82-87, 2012.

[27] A. Hatamlou, "Black hole: A new heuristic optimization approach for data clustering," *Elsevier,* p. 175–184, 2012.