ANALYSIS AND DESIGN OF AN INTEGRATED MODEL FOR INCREASING THE PERFORMANCE OF MAP REDUCE ON HETEROGENEOUS BIG DATA PROCESSING

V. Naveen Kumar[1], Dr. Ashok Kumar P S[2]

Article History: Received: 24.02.2023 **Revised:** 10.04.2023 **Accepted:** 26.05.2023

Abstract

Heterogeneous big data processing poses significant challenges due to the diverse nature of data and the varying computational capabilities of processing resources. Map Reduce is a common programme paradigm for handling massive amounts of data, but its performance on heterogeneous environments is often suboptimal. This paper presents an integrated model that aims to enhance the performance of Map Reduce on heterogeneous big data processing. The model incorporates several techniques and optimizations to efficiently utilize the available resources and minimize the impact of resource heterogeneity. Experimental evaluations determine the efficiency and advantage of the suggested model in terms of performance improvement, resource utilization, and scalability. The results specify that the integrated model can significantly enhance the performance of Map Reduce on heterogeneous big data processing scenarios. In proposed system, designed a new scheduling algorithm, Speculating Prioritize Tasks (SPT) algorithm that is very resistant to diversity. In clustering of 200 virtual machines on Elastic Compute Cloud (EC2), SPT may enhance Hadoop speed of response through a factor of two.

Keywords: Map Reduce, Heterogeneous Environment, Speculating Prioritize Tasks, Big Data, Elastic Compute Cloud.

[1]Research Scholar, Don Bosco Institute of Technology, Affiliated to Visvesvaraya Technological University,
Email: naveenvipparla@gmail.com
[2]Professor & HOD, Dept. of CSE, HKBK College of Engineering,
Email: ashokdbit2017@gmail.com

DOI: 10.31838/ecb/2023.12.s3.393

1. INTRODUCTION

MapReduce is a software development concept along with its execution that is used to development and produce huge data assortments. It is intended for scaling from an individual computer to several thousand computers, each of which may have different hardware capabilities. Heterogeneous environments are those that consist of a mix of different types of hardware, such as different types of CPUs, different amounts of memory, and different network speeds [1]. Optimizing MapReduce performance in heterogeneous environments can be challenging, but there are a number of strategies for improving performance [2].

One technique is to use a technique called task scheduling. Task scheduling is the process of assigning tasks to machines in a cluster. The purpose of task scheduling is to reduce the total time required to complete a work. There is several work scheduling methods available. and the best algorithm to use will depend on the specific characteristics of the heterogeneous environment [3]. Another technique for improving MapReduce concert in heterogeneous atmospheres is to use data locality. Data locality is the principle that data should be stored on the same machine as the code that is processing it. This can be achieved by using a technique called data partitioning as shown in fig 1.

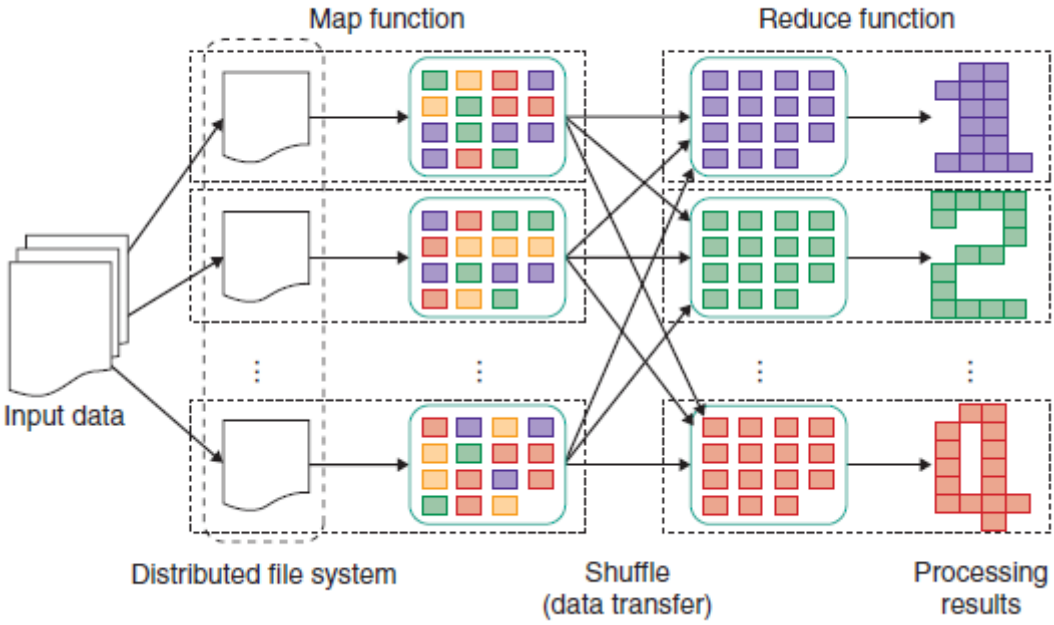

Figure 1. MapReducing Process (source: https://www.ntt-review.jp/)

Data partitioning is the process of dividing a large data set into smaller pieces, and then storing each piece on a different machine [4]. Finally, it is important to use the right tools for the job. There are a number of different MapReduce frameworks available, and each framework has its own strengths and weaknesses [5,6]. The best framework to use will depend on the specific characteristics of the heterogeneous environment.

Prioritize Tasks is a technique used to improve the act of MapReduce jobs. Scheduling works by speculatively throwing additional tasks for slow-running. The idea is that one of the speculative tasks can finish executing earlier than the original task, it may assist to minimise total work execution time [7]. Scheduling is implemented in the Hadoop MapReduce framework. When a MapReduce job is submitted, the Hadoop scheduler will speculatively launch additional tasks for any tasks that are predicted to take longer than a certain threshold [8]. The scheduler will choose the machines to run the speculative tasks on based on a number of factors, such as the machine's CPU speed, memory, and network bandwidth [9].

Scheduling has been shown to be an effective way of MapReduce jobs' efficiency should be improved. In a study conducted by researchers at the University of California, Berkeley, Scheduling was shown to reduce the execution time of MapReduce jobs by up to 50%. Elastic compute Cloud (EC2) is an Amazon Web Services (AWS) online service that offers re-sizable computing power in the cloud. It enables users to rent virtual servers, known as instances, to run applications and perform various computing tasks [10]. EC2 offers a flexible and scalable infrastructure for businesses and developers, allowing them to quickly provision and manage virtual servers based on their needs.

RELATED WORKS

Clustering enormous amounts of data is a relatively new and much discussed topic that is working in a wide-variety of uses, such as social networking, bioinformatics, and a great deal of other fields. Traditional clustering techniques need to be refined in order to accommodate the ever-increasing amounts of data that need to be managed [11]. The usefulness of the suggested HHHO was shown via the use of numerical comparisons, which demonstrated the method's effectiveness in achieving the desired results.

In addition, the current approaches lack strong methods for managing communication overhead as well as effective techniques for loading parallel data in parallel [12].

The MapReduce algorithm, which combines the map and reduces processes, is now one of the most often used categorization algorithms for huge amounts of data. The mapping strategy is used for the process of filtering and organising, and the reduction method is utilised for the purpose of combining the final classification findings [13]. The suggested method produces values for accuracy, specificity, and sensitivity for two different datasets: one including rotten tomatoes movie reviews and the other containing dermatology data.

This study presents a complete evaluation of connected subjects such as the idea of big data, modelling driven and data driven techniques, in order to completely report Big Data Amalytics (BDA) for intelligent manufacturing systems. This review is provided in addition to fully report Big Data Amalytics (BDA). The discussion focuses on the framework [14] of BDA for intelligent manufacturing systems, as well as its development, essential technologies, and application areas. Both the obstacles and the prospects for

the direction of future research are discussed. It is envisaged that new ideas will be generated as a result of this study as part of the ongoing endeavour to realise the BDA for intelligent manufacturing systems.

The suggested method [15] for clustering large amounts of data goes through two stages, which are feature selection and clustering, both of which are carried out in the first cluster nodes of architecture of the sparks.

Because the Application Programming Interfaces (APIs) for end users need to be modified in order to have access to the underlying heterogeneous hardware, the programmability of the system suffers as a result of this integration at the present time. For instance, modern Big Data frameworks like Apache Spark provide a new application programming interface (API) that integrates GPUs with the Spark programming paradigm that is already in use. In order to cluster large amounts of data, the suggested method [16] goes through two stages, which are feature selection and clustering. Both stages take place in the first cluster nodes of the spark architecture.

The inconsistent big data computing model does not take into account two levels of parallelism among the nodes. Additionally, communication related overhead is high, the running time of the model is lengthy, and the computational weight is small [17].

The resolution of this study is to develop [18] a fresh conceptual frame for BDA in the healthcare industry that is suitable for implementation in the European Mediterranean area. The big data quality (BDQ) module is a new addition to this framework. Its purpose is to filter and clean the data that are delivered from various European data sources.

Numerous studies have been conducted on velocity and volume; however there is still no comprehensive and effective solution on the market for these variables. As part of the traditional solutions offered by DBMS, multidimensional data types are often used [19]. Many of today's data formats, however, are not compatible with the conventional computer systems mentioned previously.

However, in spite of the progress made in this area of study, very few articles have been discovered that discuss architectural components. Furthermore, there is a lack of norms and the utilisation of position architectures, both of which are necessary for appropriate development [20].

PROPOSED MODEL

Hadoop considers that any node that is noticeably sluggish is broken. However, nodes might be inactive for a variety of reasons. As a result of co-location of VMs on one observable host, heterogeneity may occur in a virtualized data centre. The performance of CPUs and memories is separated by virtualization, but the capacity of discs and networks is also shared between the virtual machines. When there is no conflict, co-located VMs consume the entire bandwidth of the host and split bandwidth equitably when there is disputation. VMs belonging to other users may compete with one another temporarily, or VMs belonging to a user may compete with one another permanently, as in Hadoop. It is not fundamentally harmful to have EC2 bandwidth shared. However, this

policy does present challenges for Hadoop.

Although the scheduler utilises a set threshold to choose speculative jobs, an excessive number of speculative tasks may be started, diverting resources away from beneficial tasks. Furthermore, since the scheduler prioritises candidates based on their location, the incorrect jobs may be assigned first for guessing. For instance, if the average performance was 80% and the work was two times slower at 30% advancement and a 10 times slow task at 6% improvement, the 2 times slow than the task may be hypothesised first.

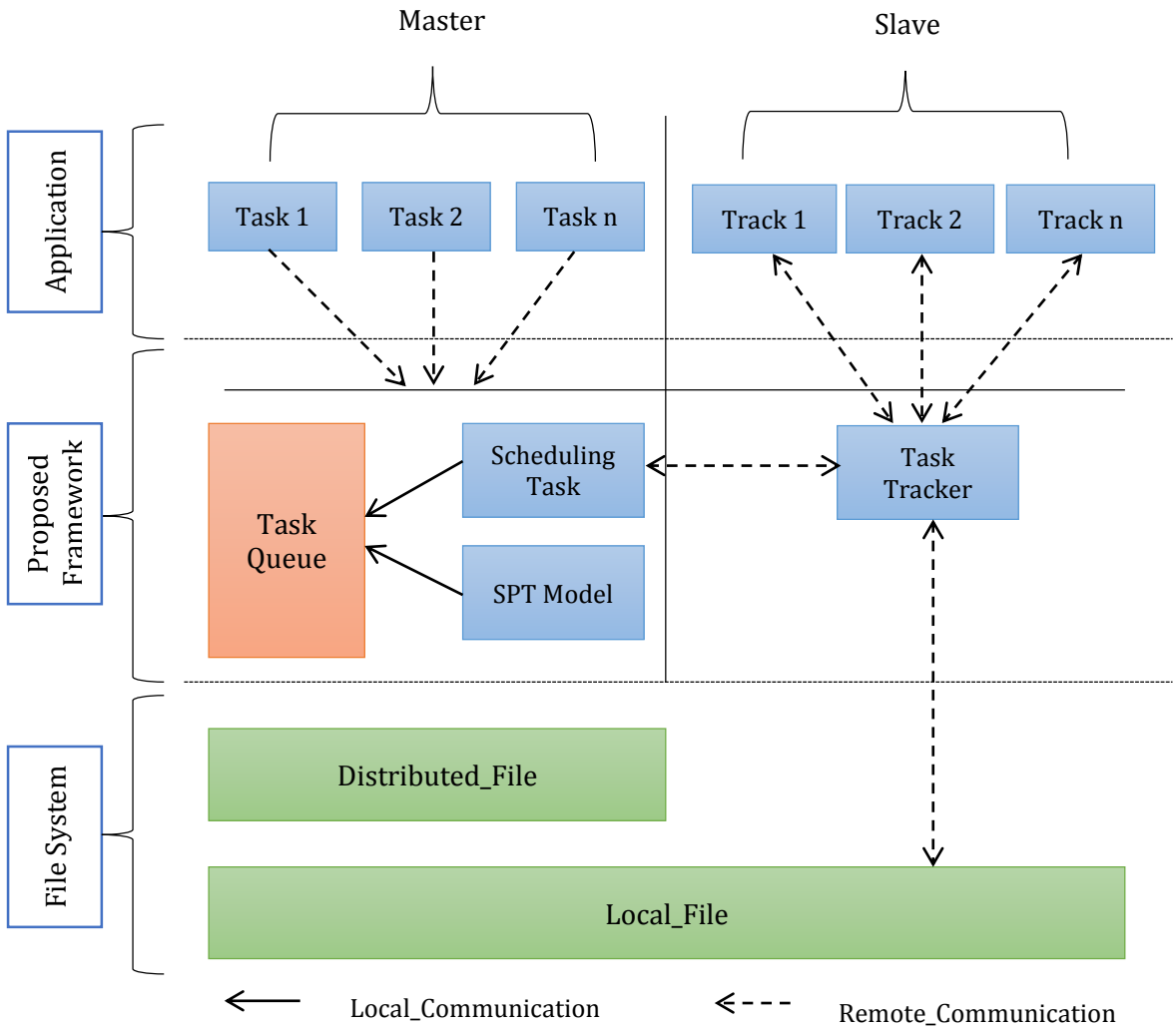

Figure 2. Architecture of Proposed Framework

The proposed SPT algorithm is depends on three principles: Prioritise tasks for speculation, choose fast nodes for execution, and limit speculating tasks to avoid thrashing. The SPT algorithm employs the following variables to realise these principles:

Slow_Node Threshold **(SNT)** - This data is the limit to prevent scheduling on sluggish nodes. This number is used

for comparing the scores of all completed and on-going jobs on the node.

Speculative_Cap **(SC)** - It is the maximum number of speculative jobs that may execute concurrently.

Slow_Task Threshold **(STT)** - It is a progress rate criterion used to assess if an activity is sluggish enough to warrant speculation. When only quick processes are running, this eliminates unnecessary conjecture.

The task's progress rate is indicated by

$$ProgressScore = ExecutionTime \quad (1)$$

The time remaining element for the work is calculated using Hadoop's Progress Score.

$$\frac{(1 - ProgressScore)}{ProgressRate} \quad (2)$$

When a node's task slot is empty, Hadoop selects a task from one of three types. First and foremost, any unsuccessful tasks are given top importance. This is done to identify and halt the job when a task fails frequently due to a problem. Second, non-running jobs are taken into account.

To identify speculative tasks, Hadoop calculates a progress score between 0 and 1. As a result of reading input data, the progress score for a map is calculated. In a reduction task, one third of the score is accounted for by each of its three components:

- Copy Phase – The job retrieves map results.
- Sort Phase - The outputs of the map are ordered by key.
- Reduce Phase – The user-defined method is assigned to each key in the set of map outputs.

While a metric's like development rate, makes greater than definitive progress for determining, a threshold in Hadoop works effectively in homogeneous circumstances due to tasks tend to begin and conclude in "waves". In conclusion, while executing many jobs, Hadoop employs a FIFO correction in which the first proposed job is requested for a task to execute, followed by the next, and so on. There is also an ordering mechanism in place to place work in higher-priority lanes.

Algorithm 1 SPT Algorithm

Step 1.A new task is requested by node N.
Step 2.If there are more than one running speculative_tasks < SC then
Step 3.If nodes are making progress <SNT then
Step 4.Delete the request
Step 5.else
Step 6.By estimated time left, rank tasks currently running without speculation
Step 7.repeat
Step 8.From the ranked list, select task T
Step 9. Assume T is progressing at a certain rate <STT then
Step 10. Activate node N by launching T
Step 11. exit
Step 12. end if
Step 13. While there are tasks on the ranked list
Step 14. end if
Step 15. end if

Let's assume we have a list of numbers and we want to analyse the sum of squares using MapReduce. The input dataset is divided into portions, and a mapper is used to process each section.

The mappers calculate the square of each number and emit key-value sequences with the key squares and the value 1. The reducer receives the key-value pairs, aggregates them by summing the values, and calculates the final sum of squares.

The mathematical model for MapReduce task job process in a heterogeneous cluster can be formulated as follows:

Let J be a MapReduce job, consisting of m tasks, and let there be n compute nodes in the cluster. Let each node be identified by a unique index i $(i = 1, 2, \dots, n)$. Let T_i, j be the execution time for task j on node i.

Then, the average and maximum task durations for a job J can be estimated as:

$$M(avg\ to\ max) = (1/m) * sum(i = 1\ to\ n) sum\ (j = 1\ to\ m)\ T_i, j \quad (3)$$

$$M(\max to\ J) = \max[sum\ (i = 1\ to\ n) T_i, j\ ] \quad (4)$$

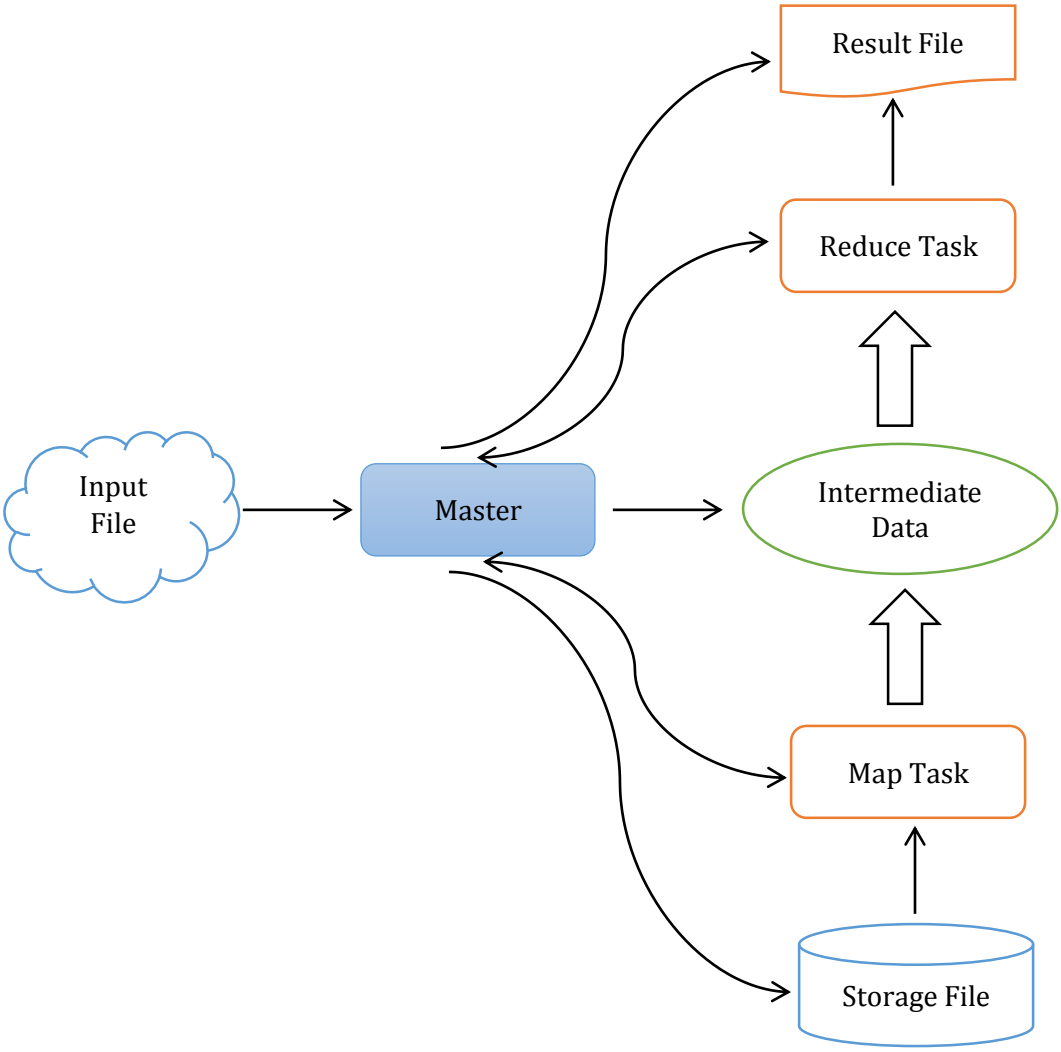

Figure 3. Map Reduce on Heterogeneous Big Data Processing

Pseudocode for Proposed Model

Map function:

```
function Map(key, value):
    // key: input chunk identifier
```

```
   // value: input chunk of numbers
   for each number in value:
      square = number * number
      emit(square, 1)
```

Reduce function:

```
function Reduce(key, values):
   // key: square of a number
   // values: list of 1s
   sum = 0
   for each value in values:
      sum = sum + value
   emit(key, sum)
```

Driver function:

```
function MapReduce(input):
   // input: list of numbers
   // Split input into chunks
   chunks = splitInput(input)
   // Map phase
   mappedData = []
   for each chunk in chunks:

mappedData.append(Map(chunk.identifier, chunk.data))
   // Shuffle and sort phase
   shuffledData = shuffleAndSort(mappedData)
   // Reduce phase
   reducedData = []
   for each key-value pair in shuffledData:
      reducedData.append(Reduce(key, pair.values))
   // Final output
   output = []
   for each key-value pair in reducedData:
      output.append(pair)
   return output
```

In this example, the Map function processes each number in the input chunk, calculates the square, and emits a key-value pair with the square as the key and 1 as the value. The Reduce function receives the key-value pairs and sums up the values to calculate the final sum of squares. The driver function orchestrates the overall MapReduce process, splitting the input, mapping, shuffling, reducing, and producing the final output.

2. RESULTS AND DISCUSSIONS

We started our analysis by assessing heterogeneity in the EC2 production system. When we did our scheduling tests, however, Amazon allocated us to a different test cluster. Amazon transferred this to cluster while our research revealed a fault in the network programme that caused connections among our VMs to fail periodically in production. Our EC2 experimentations competed on "small"-size EC2 VMs through the memory of 1.7 GB, 1 simulated core with "the equivalent of a 1.0-1.2 GHz 2007 Opteron or Xeon processor," and 160 GB of HDD on hypothetically collective hard drive.

We ran extensive tests on the SPT algorithm on an EC2 heterogeneous cluster. According to one testing, SPT completed tasks 27% quicker than It is 31% faster than Hadoop's native scheduler in a cluster with non-faulty nodes. Even in the absence of problematic nodes, SPT yields advantages in diverse settings. SPT performed Sort with stragglers tasks 58% quicker than this scheduler performs 200% quicker than Hadoop with speculative implementation disabled and 220% faster than Hadoop's native scheduler. The comparison of SPT's worst-case, best-case, and average-case performance versus the Hadoops scheduler, with no guesswork for runs without and with stragglers as shown in fig 4-8.

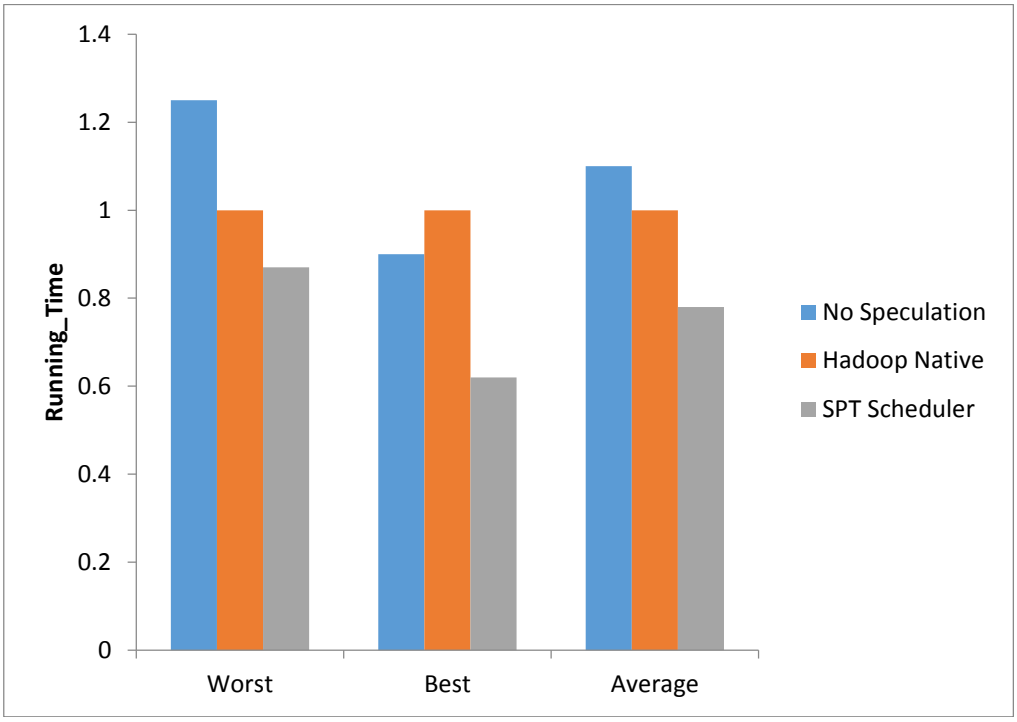

Figure 4. SPT Running Time

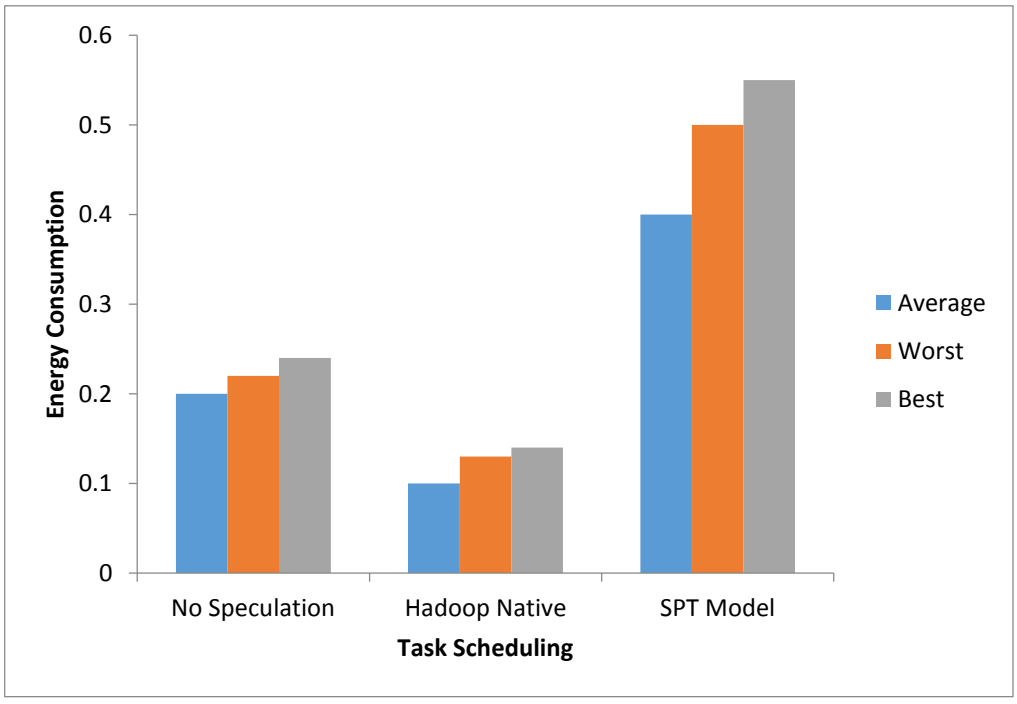

Figure 5. Energy Consumption

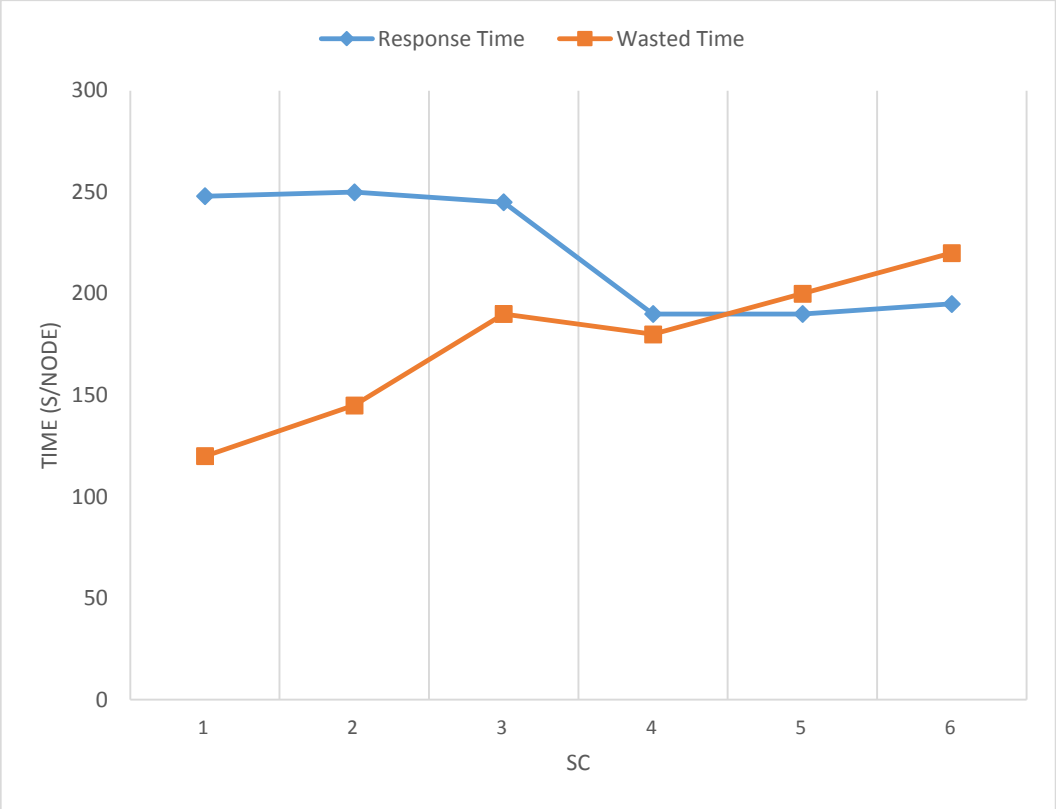

Figure 6. Performance of SC time

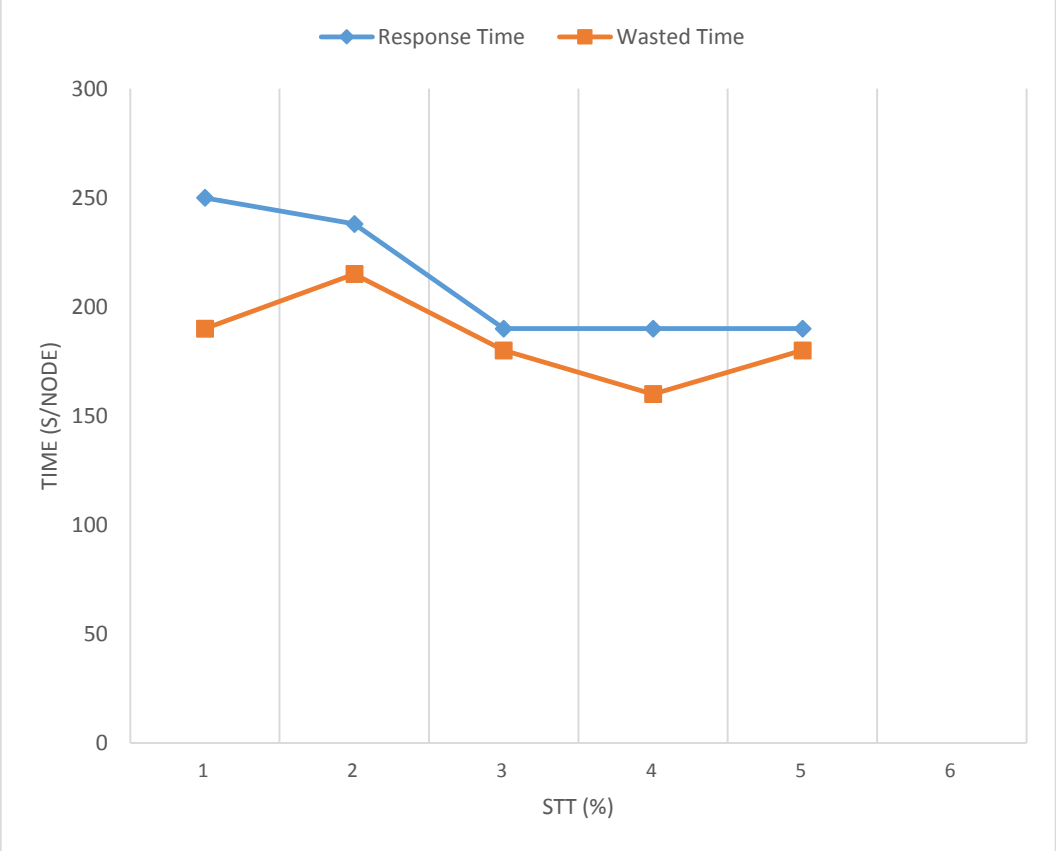

Figure 7. Performance of STT time

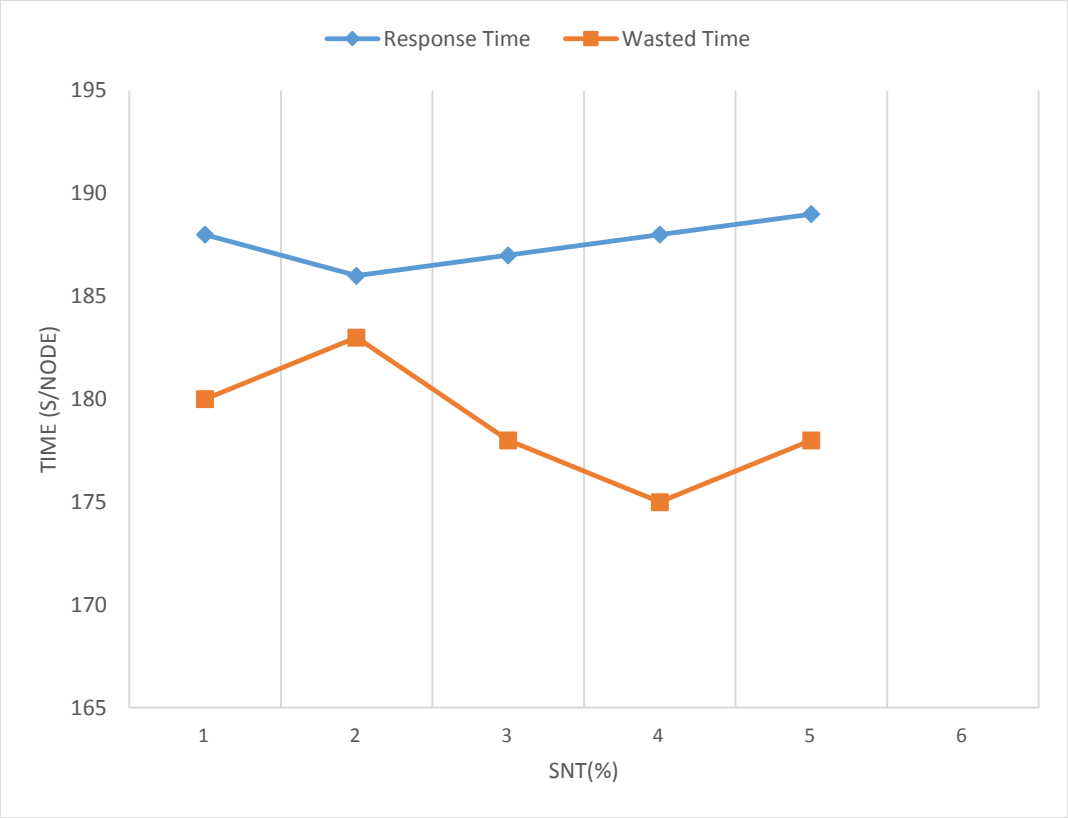

Figure 8. Performance of SNT time

In the test setting, sensitivity analysis to Speculative Cap revealed that reaction time declines substantially at Speculative Cap = 20%, after which it remains low. The greater threshold value is also undesirable since it causes SPT to spend more time on excessive speculation. Experiments with Sensitivity to Slow Task Threshold reveal that low threshold values restrict the amount of speculative tasks, whilst values over 30% function. Sensitivity study to Slow Node Threshold demonstrates that SPT functions effectively as long as Slow Node Threshold is greater than the proportion of nodes that are very slow or malfunctioning.

3. CONCLUSION

Big data processing is a critical component in extracting valuable insights from large-scale datasets. Various frameworks and tools have been developed to handle the challenges posed by big data. As data continues to grow exponentially, it is essential for organizations to adopt scalable, fault-tolerant, and efficient big data processing techniques to unlock the full potential of their data assets. In this paper, we have evaluated the performance of our approach SPT scheduling algorithm which improves the performance of hadoop. It outperforms conventional map reduce scheduling. However, SPT can also increase the overall resource usage of a MapReduce job. This is because the scheduler will be launching additional tasks, even for tasks that are not actually slow. As a result, it is important to carefully consider the trade-offs between performance and resource usage when using SPT.

4. REFERENCES

[1] Aarthee, S., & Prabakaran, R. (2023). Energy-Aware Heuristic Scheduling Using Bin Packing MapReduce Scheduler for Heterogeneous Workloads Performance in Big Data. *Arabian Journal for Science and Engineering*, *48*(2), 1891-1905.

[2] Yang, S., Jin, W., Yu, Y., & Hashim, K. F. (2023). Optimized hadoop map reduce system for strong analytics of cloud big product data on amazon web service. *Information Processing & Management*, *60*(3), 103271.

[3] Bawankule, K. L., Dewang, R. K., & Singh, A. K. (2023). Early straggler tasks detection by recurrent neural network in a heterogeneous environment. *Applied Intelligence*, *53*(7), 7369-7389.

[4] Kalia, K., & Gupta, N. (2021). Analysis of hadoop MapReduce scheduling in heterogeneous environment. *Ain Shams Engineering Journal*, *12*(1), 1101-1110.

[5] Luo, C., Cao, Q., Li, T., Chen, H., & Wang, S. (2023). MapReduce accelerated attribute reduction based on neighborhood entropy with Apache Spark. *Expert Systems with Applications*, *211*, 118554.

[6] Pandey, R., & Silakari, S. (2023). Investigations on optimizing performance of the distributed computing in heterogeneous environment using machine learning technique for large scale data set. *Materials Today: Proceedings*, *80*, 2976-2982.

[7] Jagadish Kumar, N., & Balasubramanian, C. (2023). Hybrid Gradient Descent Golden Eagle Optimization (HGDGEO) Algorithm-Based Efficient Heterogeneous Resource Scheduling for Big Data Processing on Clouds. *Wireless Personal Communications*, *129*(2), 1175-1195.

[8] Slagter, K., Hsu, C. H., Chung, Y. C., & Zhang, D. (2013). An improved partitioning mechanism for optimizing massive data analysis using MapReduce. *The Journal of Supercomputing*, *66*, 539-555.

[9] Che, D., Safran, M., & Peng, Z. (2013). From big data to big data mining: challenges, issues, and opportunities. In *Database Systems for Advanced Applications: 18th International Conference, DASFAA 2013, International Workshops: BDMA, SNSM, SeCoP, Wuhan, China, April 22-25, 2013. Proceedings 18* (pp. 1-15). Springer Berlin Heidelberg.

[10] Uma Maheswara Rao, S., & Lakshmanan, L. (2023). Security and scalability issues in big data analytics in heterogeneous networks. *Soft Computing*, 1-7.

[11] Bashabsheh, M. Q., Abualigah, L., & Alshinwan, M. (2022). Big data analysis using hybrid meta-heuristic optimization algorithm and MapReduce framework. In *Integrating meta-heuristics and machine learning for real-world optimization problems* (pp. 181-

223). Cham: Springer International Publishing.

[12] Babar, M., Jan, M. A., He, X., Tariq, M. U., Mastorakis, S., & Alturki, R. (2022). An optimized IoT-enabled big data analytics architecture for edge-cloud computing. *IEEE Internet of Things Journal*.

[13] Chidambaram, S., & Gowthul Alam, M. M. (2022). An integration of archerfish hunter spotted hyena optimization and improved ELM classifier for multicollinear big data classification tasks. *Neural Processing Letters*, *54*(3), 2049-2077.

[14] Wang, J., Xu, C., Zhang, J., & Zhong, R. (2022). Big data analytics for intelligent manufacturing systems: A review. *Journal of Manufacturing Systems*, *62*, 738-752.

[15] Ravuri, V., & Vasundra, S. (2020). Moth-flame optimization-bat optimization: Map-reduce framework for big data clustering using the Moth-flame bat optimization and sparse Fuzzy C-means. *Big Data*, *8*(3), 203-217.

[16] Xekalaki, M., Fumero, J., Stratikopoulos, A., Doka, K., Katsakioris, C., Bitsakos, C., ... & Kotselidis, C. (2022). Enabling Transparent Acceleration of Big Data Frameworks Using Heterogeneous Hardware. *Proceedings of the VLDB Endowment*, *15*(13), 3869-3882.

[17] Yin, F., & Shi, F. (2022). Heterogeneous Big Data Parallel Computing Optimization Model using MPI/OpenMP Hybrid and Sensor Networks. *ACM Transactions on Sensor Networks*.

[18] El Samad, M., El Nemar, S., Sakka, G., & El-Chaarani, H. (2022). An innovative big data framework for exploring the impact on decision-making in the European Mediterranean healthcare sector. *EuroMed Journal of Business*.

[19] Naeem, M., Jamal, T., Diaz-Martinez, J., Butt, S. A., Montesano, N., Tariq, M. I., ... & De-La-Hoz-Valdiris, E. (2022). Trends and future perspective challenges in big data. In *Advances in Intelligent Data Analysis and Applications: Proceeding of the Sixth Euro-China Conference on Intelligent Data Analysis and Applications, 15–18 October 2019, Arad, Romania* (pp. 309-325). Springer Singapore.

[20] Cravero, A., Bustamante, A., Negrier, M., & Galeas, P. (2022). Agricultural Big Data Architectures in the Context of Climate Change: A Systematic Literature Review. *Sustainability*, *14*(13), 7855.