ANN-inspired Straggler MapReduce Detection in Big Data Processing

Ajay Bansal · Manmohan Sharma · Ashu Gupta

Abstract One of the most challenging aspects of using MapReduce is detect straggler nodes while processing large-scale data in parallel manner. Identifying ongoing tasks on weak nodes is how it's described. The overall calculation time is the amount of time taken by the system which is categorized into two phases; Map Phase (duplicate, join) and the Reduce Phase (mix, sort, and lessen). The primary objevtive of this study is to estimate the accurate execution time in each location. The proposed approach uses an Artificial Neural Network (ANN) with Genetic Algorithm (GA) on Hadoop to detect straggler tasks and calculate the remaining task execution time, which is crucial in straggler task identification. The comparative analysis is done with some efficient models in this domain, like LATE and ESAMR. The actual execution time for Word-Count is evaluated and benchmarking is done. It was found that the proposed model is capable of detecting straggler tasks in accurately estimating execution time. It also helps in reducing the execution time that it takes to complete a task.

Keywords MapReduce · Hadoop · Straggler Tasks · Speculation · Artificial Neural Network

1 Introduction

Daily life of individuals is now a days are highly becoming dependent on Information technology [1,2]. In 21 century, enormous amount of data [2] has been generated by various IT enabled products like IoT (Internet of Things) which include medical equipment, RFID tags, logs generated by various business or similar applications, scientific research data and so on. A lot of study [4–6] by researchers have been done to handle and organize these data which resulted in various data modeling techniques [8] and query handling mechanism by splitting the query and running it parallelly on various nodes which host the data. In 2003, Hadoop [12] was introduced as an free available software framework to handle, organize and process the large amount of data spread across various data centers or nodes hosted locally or distributed. In Hadoop, data is organized by HDFS (Hadoop Distributed File system) [3] and processed with the help of a powerful framework called MapReduce [8–10]. In 2010, a refined resource manager and negotiator YARN (Yet Another Resource Negotiator) was introduced to control the ever-increasing demand of resources to handle the current running tasks.

As a part of the Big Data processing infrastructure [8], the MapReduce method divides the process into the number of modules known as tasks. By utilizing a traditional database management system like Oracle, SQL Server and others well known RDBMS softwares, it is difficult to store and analyze large volume of data (mix of structured or unstructured) is difficult. However, million of clients can track down their data across the web in a couple of seconds due to MapReduce architecture [9]. MapReduce process is divided into two phases. First phase is known as the Map stage while the subsequent stage is known as the Reduce phase. Mapping is the operation which is used to read, sort and combine the input data. In the MapReduce framework [10], tasks are initially delivered to the machine for mapping, that conducts two phase: combine and copy. The overall execution time is the amount of time taken by all the steps. Every stage has a weight assigned to it and the weight is calculated as a ratio of the execution time of the phase divided by the total execution time of the task. Since data is transported during most of the duplicate and mix stages, hence they have the most impact on execution. The

Ajay Bansal
Lovely Professional University, Jalandhar, 144411, Punjab, India
E-mail: ajayg13@rediffmail.com

Manmohan Sharma
Lovely Professional University, Jalandhar, 144411, Punjab, India
E-mail: manmohan.sharma71@gmail.com

Ashu Gupta
Prince Sattam Bin Abdulaziz University, Saudi Arabia
E-mail: guptashu1@rediffmail.com

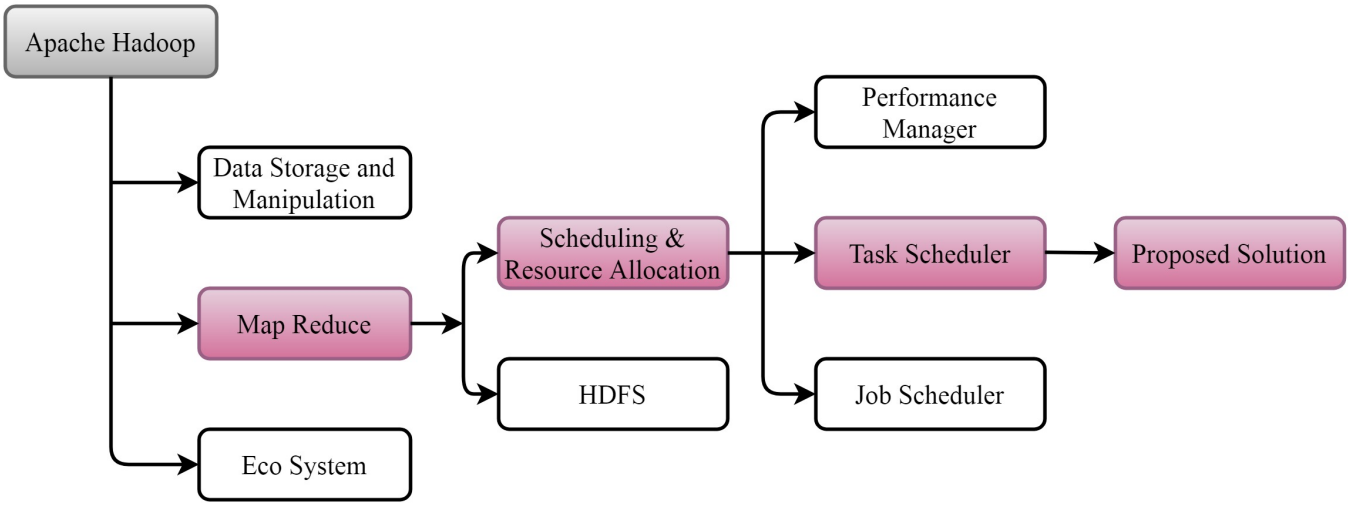


Fig. 1 The proposed straggler prediction sequence in Hadoop framework

absolute time taken to complete the whole job is determined by the speed at which the slowest assignment is finished.

A straggler [11] task is basically a task which is executing on the slowest node and that need to be identified and allocated to another node using speculative execution [16]. Given the heterogeneous nature of Big Data processing, assigning the same weight to each stage is inefficient. The K-means algorithm [27] is used by ESAMR [18] to measure the weights of each task. A dynamic technique for calculating completion time is created in the proposed study, which may be employed in both homogenous and heterogeneous contexts. On the basis of task processing time, the values are determined using neural networks. This study attempts to increase the Big Data processing infrastructure's effectiveness by lowering the inaccuracy of determining time duration for task completion by identifying straggler tasks in proactive manner and executing speculative task in place of straggler task well in advance [14].

One of the most challenging aspects of using MapReduce is to parallelize and distribute large-scale data processing for detecting straggler tasks [7]. The total computation duration is the total execution times of the two stages known as Mapping (copy, combine) and Reducing (sort, shuffle, and reduce). The major goal of this research is to determine the exact completion time at every site. To identify straggler jobs and determine execution time, the suggested solution leverages a backpropagation system on Hadoop. For the WordCount, the comparative analysis is carried out using several efficient models in this area, such as ESAMR [18] and LATE [19]. It was found that the proposed model is capable of detecting straggler tasks so that execution time can be estimated accurately. It also helps in reducing the execution time that it takes to complete a task.

Article Structure: The rest of the article is structred as follows. Some imperative work related to the proposed study is discussed in Section 2. The proposed solution is discussed in Section 3. The calculated experiments are discussed in Section 4. Lastly, the article is concluded in Section 5.

2 Backgroud Work

The proposed approach is focused on addressing the flaws in existing literature-based solutions. For example, in algorithm called LATE, the rest of the time for each step of the running task is treated the same. On the other hand, Reduce phase takes longer time to complete the task compared to the other stages. Since they are dependent on the previous mission, the SAMR [20], ESAMR, and SECDT [22] methods cannot estimate the running time accurately. It is not precise enough due to the differences in characteristics between previous and current tasks. ESAMR only considers executable information and ignores node requirements, which is necessary because node processing times vary depending on their feature such as memory and CPU. For efficient data transmission, the authors proposed a comprehensive transmission (CT) model. They additionally devise a two-stage asset sharing (TPRS) convention, which normally comprises of a pre-separating stage and a check stage, to achieve allowed asset partaking in the CT model effectively.

2.1 Hadoop Native method

To control the nodes in the Hadoop, a method called Hadoop Native Algorithm is proposed. In the proposed technique, the weights are denoted by m1=0, m2=1 and ra1=ra2=ra3=1/3, appropriately. After the task has been completed for at least 1 minute, the following calculations are performed. The progress score of (P_s) of operation in Map and Reduce stage in the range of (0,1) and mathematically calculated in Equation 1 and 2.

$$P_s = \frac{X}{Y} \tag{1}$$

$$P_S = \frac{1}{3}(K + \frac{X}{Y}) \tag{2}$$

The quantity of matches that have been effectively handled is X, and the all out number of matches is Y. In a Reduce stage, K is the stage (shuffle, sort, and merge). The average value of (P_s) is calculated in Equation 3 assigned to the threshold, where N is denoted as the total number of executing tasks.

$$Avg(P_s) = \sum_{i=0}^{N} p_s[i]/N \tag{3}$$

Finally, Equation 4 detects the straggler node by ensuring that the value of P_s is less than 20% of the average (P_s).

$$P_s \leq Avg(p_s) - 20\% \tag{4}$$

2.2 Longest Approximate Time to End (LATE) approach

LATE aims to figure out how much time is left to complete tasks. While the loads are equivalent to the Hadoop naive interaction. Equation 5 considers the remaining time when selecting straggler tasks.

$$pr = \frac{P_s}{t} \tag{5}$$

Pr denotes the progress rate, and the task consumed time is indicated by t. Equation 6 is used to determine how much time is left to complete tasks in a given stage (TTE).

$$TTE = \frac{(1 - P_s)}{Pr} \tag{6}$$

The tasks are sorted by the amount of time they have left. A hypothetical cap of 10% of completed tasks is used to choose straggler tasks. They will be reassigned to a node that meets the requirements of Equation 7.

$$Slow\ node\ threshold = 25\%\ of\ all\ nodes \tag{7}$$

LATE has no way of knowing how much time is left. It involves steady numbers for each progression's weight, despite the fact that the effect of each progression for each assignment fluctuates, and the proportions are not consistent all of the time. The proposed method finds related tasks and determines acceptable weights for each move using stored executable information.

2.3 Self-adaptive MapReduce Scheduling Algorithm method (SAMR)

This technique saves the final weights of finished jobs at each point in a XML document and uses it to anticipate the following process execution time. The consequences for the first stage are (1,0,1/3,1/3,1/3). Every 100 milliseconds, Ps is measured. Equation 8 calculates the average execution speed, while Equation 9 identifies the straggler job.

$$APR = \sum_{i=1}^{N} \frac{Pr[i]}{N} \tag{8}$$

$$Pr[i] < (1 - STaC) \times APR \tag{9}$$

Straggler tasks are indicated by the closeness of STaC to zero. Equation 10 is often used to measure the number of backup tasks denoted as BackupNum, where the number of executing taks as represented as TaskNum.

$$BackupNum < B_p \times TaskNum 0 < B_p < 1 \tag{10}$$

Bp has been calculated to be 0.2. Equation 11 calculates the average completion time of all running tasks (ATTE).

$$ATTE = \sum_{k=0}^{N} \binom{n}{k} \frac{TTE[i]}{N} \tag{11}$$

It is a slow task if it fulfills Equation 12.

$$TTE[i] - ATTE > ATTE \tag{12}$$

The Slow Task Threshold (STT) is a scale that measures how fast or slows a task is in the range [0,1]. We use STT = 0.4 in this article since STT less than 0.4 considers most of the fast tasks to be slow. Many slow tasks are expected to be quick when STT is greater than 0.4.

2.4 Enhanced Self-adaptive MapReduce Scheduling Approach (ESAMR)

In 2012, ESAMR was implemented to enhance the capcbility of SAMR algorithm. The algorithm divides the total knowledge into k number of groups and stores weights from previous tasks. A temporary weight will be determined after compppleting Map and Reduce stages. The calculation searches for a bunch with the most reduced load to appoint the to some extent finished jobs to them. If a node does not have any completed tasks, the average weight of all collections will be utilized.

3 Proposed Model

There are two parts to the execution by utilizing ANN system. Intially, the suggested method may be evaluted in both ”heterogeneous” and ”homogeneous” sourroundings [24]. Secondly, it precisely calculates each level’s values, resulting in a reduced overall processing time. The suggested method employs two types of variables: (values allocated on every part and rest of the MapReduce task processing period) and independent (amount of processed data, progress rate, and execution time) variables. The proposed approach has three key components, as shown in Fig. 2 a file storage database for storing data from older performed process, estimation of weight by the Deep learning, andtask execution time estimation. Each new task will be separated into 2 phases: Map and reduce. As input data for NN to calculate values, the produced outputs are divided into training and testing sets databases. The following tests may also be used to build the machine and increase the validity of the information.

A flowchart depicts the correlation among the multiple components shown in Figure Fig. 2. The resource manager can devote staff to this job if you join a job through Hadoop. After t seconds, the machine manager looks for straggler tasks. In this stage success rate, progress score, and residual processing time for each task will be determined using executing information from working nodes in order to identify straggler tasks. The remaining execution time is used to sort the tasks. If the amount of speculating tasks exceeds 10% of all activities, the work by far the most time left will be completed. The client can allocate a job to Hadoop using the suggested form, which causes the system manager to devote resources to the new job. The system manager will search for straggler tasks after each task has run for t seconds. The next move is to rank all of the remaining tasks according to how much time they have left to complete them. Slave nodes save information about existing jobs that are being run speculatively. The network administrator gathers all data about presently operating processes and uses the NN technique to process it. If the tasks are greater than 10% overall tasks, the success rate, remaining processing time, and task progression are determined. In that case, no modifications to the execution procedure will be made. Otherwise, the task with the least amount of time left would be started. This procedure will be followed for all new hires. In this study, we estimated weights using a backpropagation neural network. After each job is done, the weight of the stage is computed by dividing the total completion time of each phase by the processing time of every phase (in Reduce and Map). This recorded readable knowledge is used by the neural network to predict runtime.

3.1 Scheduler

The suggested approach is intended to enhance the process of speculative execution. A freshly chosen job’s tasks will be divided into MapReduce jobs and allocated to a node. The source participant received and stores the objective execution information on a regular basis. On every node, we employ a capsule to conduct specified tasks. The resource management checks up on the jobs while they are running to check if there are any stragglers.

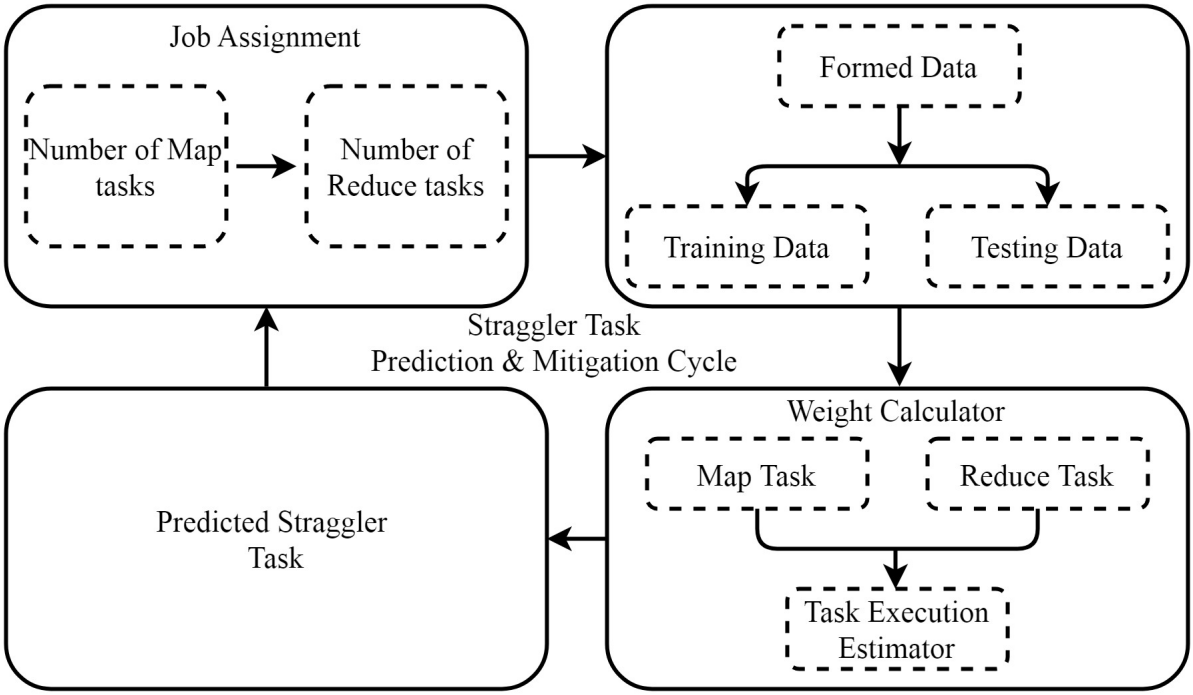


Fig. 2 The Structure of the proposed framework

3.2 Nural Network Weight calculation

The value of real weight further compared to the approximate one in each iteration of the proposed process. The neural network calculates the weights of all three stages of the reduction process. In this study, backpropagation NN is used, and the approximate weight is compared to the actual one in each iteration. The learning can either move on to the next cycle or halt at the present guesses. P_s will be determined using Eq. 13. Mathematically, the MAP is represented as;

$$P_s = \frac{x}{y} \tag{13}$$

Here, P_s defines the score of the progress made by the nodes. In this manner, x defines the amount of process data and y defines the amount of total data that need to be processed. After processing the data, weights of the networks are optimized by following the principles of GA. In the field of computing, GA is considered one of the most influential heuristic search techniques [25]. GA is a transformative calculation that utilises the principles of biological sciences like "inheritance", "selection", "crossover", and "mutation". The benefits presented by these strategies are framing a superior ANN-GA framework that can further develop speculation and simultaneously facilitate the ANN configuration process. In the proposed study, GA is utilized to extricate the significance of a singular element in the dataset [26]. Also, two distinct arrangements of component significance that depend on normal and best order precision have been laid out. In view of the best and normal element significance, the work has been stretched out by some series of examinations by manual determination of component subsets by following the concept of GA. The principal highlight subset is made by embedding the main position include, trailed constantly include a subset that comprises of first and second position highlights. The final part of the dataset comprises all highlights. Thusly, a decent component subset that may be missed by GA can be physically investigated.

3.3 Reduce phase

The estimated weights of every level are used to estimate the total and remaining execution time. P_s and completion duration are controlled by the quantity of data that can be collected and the existing job completion stage. The work that takes the slowest to complete is allocated to a different node for query execution. Stale jobs aren't rerun; instead, they're rearranged according to the available runtime. The proposed approach treats the mappings as a single stage, thus the neural network containing P_s and x outputs, as well as the remainder computational efficiency, is generated and saved in the TTE. Mathematically, the REDUCE phase is calculated into three stages as follows:

$$PS = R_1 \times P_s \tag{14}$$

Table 1 Experimental Parameter and constant

Experiments	Static Parameters	Dynamic Parameters
1	Calculated weight from the traning dataset	Comparing the estimated weight
2	Estimated weight with the assistance of a ANN-GA calculation in Reduce Phase, Progress pace of executed undertakings	Assessed weight with the assistance of an ANN-GA calculation in Map stage
3	Weight of map-reduce phase obtained from experiment 2	Remaining time of tasks' execution
4	Weight of executed tasks related to map-reduce phase	Number of hubs, Amount of info source administrator after 60s of execution look about the straggler task.

$$PS = R_1 + R_2 \times P_s \tag{15}$$

$$PS = R_1 + R_2 + R_3 \times P_s \tag{16}$$

Here, PS defines the score of the progress made by the node which are executing the REDUCE tasks. PS is calculated for Stage 1, 2, or 3 of the REDUCE task. Eq. 14 calculates PS for the REDUCE task which are currently running in Stage 1 and R_1 is weight for Stage 1. Similalrly, Eq. 15 for Stage 2 abd Eq. 16 for Stage 3 where R_2 is the weight of Stage 2 and R_3 is the weight of Stage 3.

4 Performance Evaluation

On each virtual machine, the proposed approach is implemented using Hadoop 2.7.3 and Ubuntu 16.0. (VM). The version of Hadoop conjecture section has been updated. The simulation computer has 16 GB of RAM and a 3.2 GHz Intel i7 processor. We examined two slaves for each of the Reduce and Map jobs. The findings are generated using the t-word counting programme, which is based on task runtime, task completion time, and phase weights [12] and compared to No-Speculate, LATE, and ESAMR. WordCount reads text files and counts how many times each word is repeated. Mapping is in charge of distributing number lines to multiple words and generating key/value pairs for each letter. The distributed method has been followed to install Hadoop clusters. A total of 5 number of cluster nodes have been considered with the level of RAM with 3G and 4G. The system is contaning the virtual machines with the storage capacity of 50GB. The data node and the MapReduce tasks are performed on the local node. The distribured file system is having the block size of 265MB.

A network controller, a job tracker, a supervisor, and 4 slave nodes make up the clusters. Each slave has an information node and a task responder. A data node and a mission follower are located on each slave. To test faults in weight estimate, task completion time, and total completion time, we conduct four independent operations. "A comparison of the proposed method and SVR is included in the first experiment". The second experiment uses data from the execution process to measure weights for the Map and Reduce phases. The remaining execution time is calculated in the third experiment using the weights that have been obtained. The final experiment will assess the impact of speculative execution on execution time as a function of data volume and the number of nodes as shown in Table 1.

4.1 Experiment 1

For forecasting the continuous nature of weights, support vector networks and artificial neural network (ann are appropriate regression algorithms. The values are defined based on error minimization in the proposed method. In contrast, SVR converts the risk of incorrect classification into an objective function and adjusts and optimizes the parameters on target function. In this part, the proposed method's results are compared to decision tree and SVR. The below equation is used to measure errors. The proposed method outperforms SVR by 79% and the decision tree method by 61%.

$$Error_{methods} = \frac{1}{N} \sum_{i=1}^{n} e_i^2 \tag{17}$$

Table 2 Weight calculation using different algorithms

LATE Algorithm		ESAMR		Proposed Solution	
Real values	**Calculated value**	**Real values**	**calculated value**	**Real values**	**Estimated value**
0.88	0.79	0.56	0.91	0.74	0.920
0.303	0.191	0.64	0.923	0.449	0.541
0.32	0.213	0.44	0.398	0.616	0.731
0.139	0.148	0.35	0.138	0.235	0.423
0.98	0.95	0.26	0	0.251	0.499

Table 3 For the second experiment, weight estimation using various approaches vs real weights of a Word Count

Proposed method		ESAMR		LATE	
SG1	SG2	SG1	SG2	SG1	SG2
0.88678–0.80341	0–0.0001	0.9543–0.8436	0–0.033	0.33–0.6442	0.33–0.09072
0.7–0.75	0–0.002	0.56–0.789	0–0.0002	0.33–0.0001	0.33–0.0514
0.8025–0.9871	0.064–0.0749	0.952–0.8521	0.0439–0.0356	0.33–0.75	0.33–0.0899
0.876–0.8544	0.00088–0.0001	0.6471–0.799	0.0057–0.0024	0.33–0.002	0.33–0.06529
0.9645–0.86	0.09158–0.0922	0.8531–0.765	0.055–0.042	0.33–0.849	0.33–0.0603
0.90467–0.9048	0.064–0.069	0.739–0.8982	0.074–0.059	0.33–0.994	0.33–0.06078

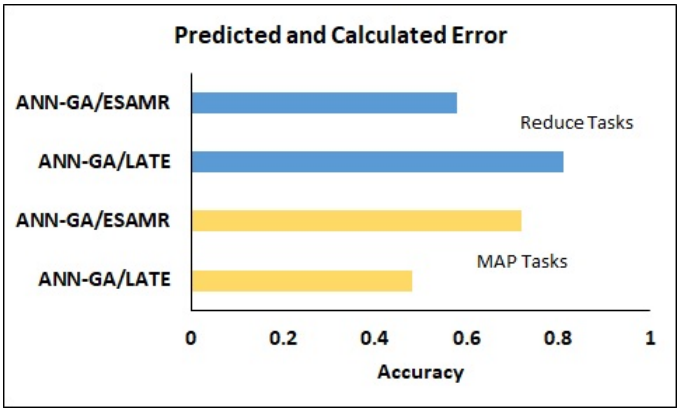


Fig. 3 Comparision of error handling with sate-of-the-art

4.2 Experiment 2

There are two parts to the mapping process and three parts to the reduce phase. In this part, we use the presented methods, LATE and ESAMR, to compare the approximate weights in these two phases. For this portion, the stored executive data is captured in a container and trained on data using Neural network, including the volume of executed information and exported to which activities are completed. The proposed technique uses backpropagation to find weights, while the ESAMR algorithm uses a value of k of 10. Compared to ESAMR, the results show an 87% increase and a 96% improvement compared to the LATE. Table 3 compares the proposed method's weight calculation to the ESAMR and LATE algorithms. Each cell in this table contains two numbers. The real weight is represented by one, and the estimated weight by the algorithm is represented by the other. These two numbers must be as similar as possible to improve the method's accuracy by correctly identifying straggler tasks.

4.3 Experiment 3

The purpose of fault injection is to shorten the time it takes to complete a task. As a consequence, the previous time estimation in the provided approach is taken into account in this experiment. The initial step was to store the results of using various input values to run the word count software. Then 20 jobs (map and reduce) are chosen to determine the estimated execution time. ESAMR [30] is tested on six computers, one of which serves as the manager and the others as running nodes. Studies for this paper are carried out on a Hadoop cluster of five computers (four working nodes and one manager). calculated time and the procecssing time for "Map and reduce" are shown in Figure 5 and Figure 6.

The suggested method, among other ways, is the nearest instance and has the most accurate estimation of the real duration and the quickest duration. According to Figure Fig. 3, the proposed approach has a 56% improvement in error rate compared to ESAMR and a 81% gain compared to LATE. The primary benefit of our approach over current methods is the accurate calculation of task remaining time. For the Map phase of ESAMR, LATE, and our process, the minimum distance among the actual and projected weights is 69.3 43.15 20.4 and for the Reduce phase, it is 186.35 99.35 37.4, respectively. ESAMR is more accurate than LATE, and we've included in the Appendix to demonstrate how effective the

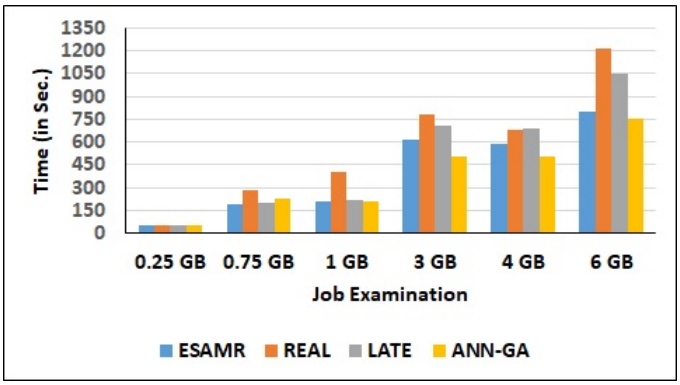
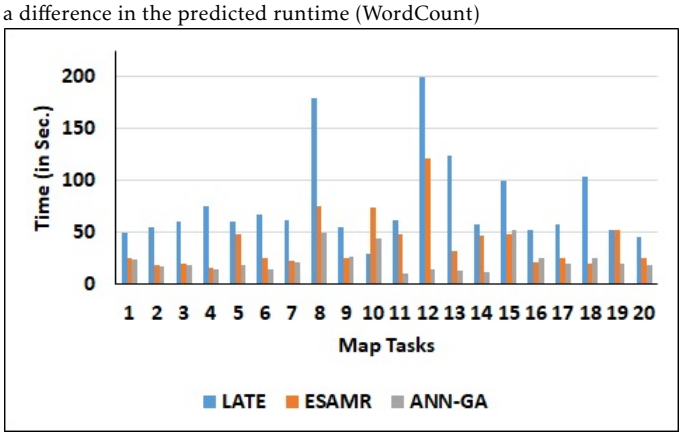


Fig. 4 In the reduction step, there is a difference in predicted run time (WordCount)

Fig. 5 In the mapping step, there is a difference in the predicted runtime (WordCount)



suggested technique is compared to ESAMR in Word Count. It calculates the difference between ESAMR's and our method's estimation error for each job in Map and Reduces.

4.4 Experiment 4

In this experiment, the impact of information and the amount of devices in the network on runtime is explored. For this purpose, the word limit programme was ran with input variables under the same circumstances for all three methods. Figure 6 shows the results obtained with two slave nodes; Figure 7 shows the results obtained with 3 nodes as slave, and Figure 8 shows the outcome obtained using 4 nodes as a slacve. Expanding the number of networks does not inevitably lead in a quicker runtime, as demonstrated. Because of the longer information time conducted between the nodes from transfering information, adding nodes in high-volume data is economical, while adding nodes in less-volume data does not increase or alter the processing time. The proposed method's remaining execution time is estimated to be 37.5% faster than real-time and 17 percent faster than "ESAMR". By recognising jobs early and allocating them to another node, the processing time is minimized. Fig. 9 tabulate the imrovements in the performed experiments.

4.5 Qualifying the proposed method by Utilizing Sort benchmark

Figures 10 and 11 contrast the time with finish assessment mistake of Map and Reduce phase utilizing ESAMR, LATE, and the proposed procedure on a Sort 6 GB work. The suggested technique beats the ESAMR in the large majority of jobs, despite the fact that the ESAMR has a tiny inaccuracy on diverse tasks. The differences in anticipated and actual time to perform "Map and Reduce" tasks using ESAMR are 4 and 9 seconds, correspondingly, but our technology's error is 5 and 8s. The average gap among the actual and predicted weights for the Map phase of ESAMR, LATE, and the suggested process is 21.3, 3.07, 2.91, and for the Reduce phase, it is 131.08, 9.29, 9.21. ESAMR delivers findings that are comparable to those of the model which we have presented. We have determined the difference between ESAMR's estimation error and the proposed method's calculated error for each job in both Reduce and Map.

ISSN: 0010-8189

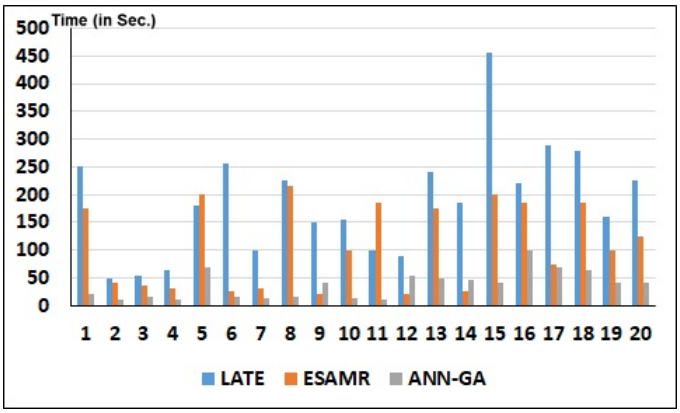


Fig. 6 Run time with two slaves

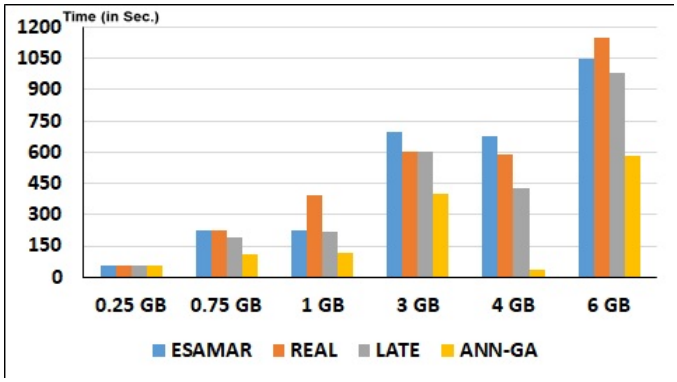


Fig. 7 Runtime with three slaves

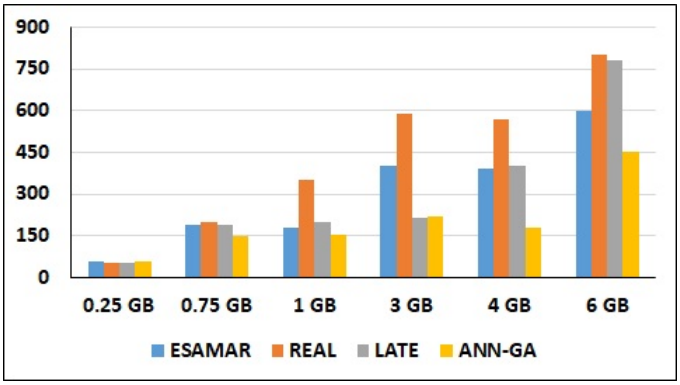


Fig. 8 Four slaves on runtime

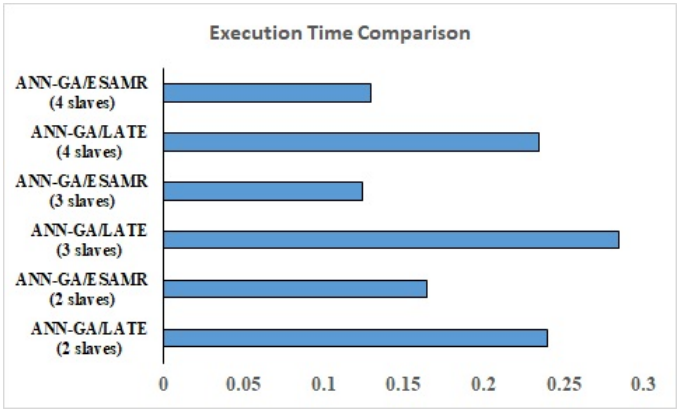


Fig. 9 When comparison to the neural net algorithm implementation strategy and LATE ESAMR, there is a percent improvement.

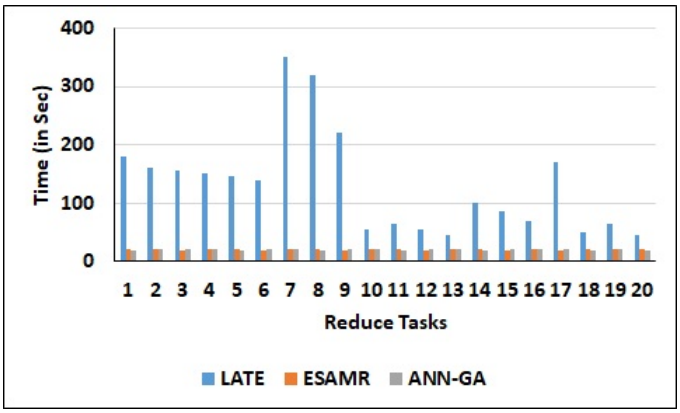


Fig. 10 In mapping phase variations in estimated runtime (Sort)

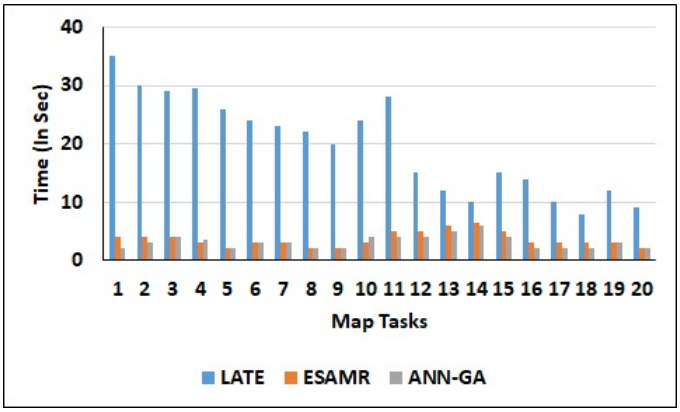


Fig. 11 In reduce phase vairation in estimated run time (Sort)

5 Conclusion

The speed with which data is analyzed is critical in Big Data processing. This research aims to improve the efficiency of Big Data's computing infrastructure to speed up data processing by identifying straggler tasks through speculative execution. "Speculative execution" using the the technique called ANN is proposed to achieve this aim. The value of weight estimate enhanced by 87.2 percent with comparision to "ESAMR" and by 96 percent compared to the LATE as a result of this growth. In addition, as compared to ESAMR and LATE, the proposed approach reduced execution time by 34% and 19%, respectively. Considering the significance of quick processing time, the proposed technique can be enhanced in the future. An automatic approach is employes to pick the best node for doing a straggler job to enhance the anticipatory process flow. To estimate the input data, additional data, such as the number of errors in various stages, might be used. The recommended approach begins the assistance task at the outset, and the performance can be speed up by maintaining the straggler labour. Before assigning a task, it's a good idea to assess the number of previous node failures. In the future, a mix of intuitive and artificial computational intelligence techniques may be used to examine data at the same time and, as a result, perform jobs more quickly. We calculate the remaining task time, identify the task for straggler, and calculate value in this article using NN. Other prediction approaches such as learning automata, Taylor, and reinforcement learning should be used in future work.

References

1. Gordon Bell, Tony Hey, Alex Szalay (2009) "Beyond the data deluge", Science 323 (5919) p. 1297–1298.
2. Tony Hey, Stewart Tansley, Kristin Tolle (2009) "The fourth paradigm: data-intensive scientific discovery", Microsoft Research.
3. White T (2009) "Hadoop : the definite guide", 1st edn. OReilly Media Inc, Sebastopol.
4. Begoli E, Horey J (2012) "Design principles for effective knowledge discovery from big data", Proceeding of the joint working IEEE/IFIP conference on software architecture (WICSA) and European conference on software architecture (ECSA), p. 215–218.
5. O'Driscoll A, Daugelaite J, Sleator RD (2013) ""Big Data", Hadoop and cloud computing in genomics", J Biomed Inform 46(6) p. 774–781.
6. Xia Z, Kai K, YuZhong S, Yin S, Minhao X, Tao P (2013) "Insight and Reduction of Map Reduce Stragglers in Heterogeneous Environment", IEEE.
7. Demchenko Y, Grosso P, de Laat C, Membrey P (2013) "Addressing Big Data Issues in Scientific Data Infrastructure", International Conference on Collaboration Technologies and Systems (CTS). IEEE Computer Society.
8. Katal A, Wazid M, Goudar RH (2013) "Big data: issues, challenges, tools and good practices", Sixth international conference on contemporary computing (IC3), p. 404–409

9. Dean, J., & Ghemawat, S. (2004). MapReduce: Simplified data processing on large clusters.
10. Bharath, R. Map Reduce: Data Processing on large clusters, Applications and Implementations.
11. Phan, T. D., Pallez, G., Ibrahim, S., & Raghavan, P. (2019). A new framework for evaluating straggler detection mechanisms in mapreduce. ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS), 4(3), 1-23.
12. Polato, I., Ré, R., Goldman, A., & Kon, F. (2014). A comprehensive view of Hadoop research—A systematic literature review. Journal of Network and Computer Applications, 46, 1-25.
13. Pol, V. V., & Patil, S. M. (2016, August). Implementation of on-process aggregation for efficient big data processing in Hadoop MapReduce environment. In 2016 International Conference on Inventive Computation Technologies (ICICT) (Vol. 3, pp. 1-5). IEEE.
14. Sakr, S., Liu, A., & Fayoumi, A. G. (2013). The family of mapreduce and large-scale data processing systems. ACM Computing Surveys (CSUR), 46(1), 1-44.
15. Badita, A., Parag, P., & Aggarwal, V. (2020). Optimal server selection for straggler mitigation. IEEE/ACM Transactions on Networking, 28(2), 709-721.
16. Chen, Q., Liu, C., & Xiao, Z. (2013). Improving MapReduce performance using smart speculative execution strategy. IEEE Transactions on Computers, 63(4), 954-967.
17. Sun M, Zhuang H, Li C, Lu K, Zhou X (2016) Scheduling algorithm based on prefetching in mapreduce clusters. Appl Soft Comput 38(C):1109–1118
18. Sun, X., He, C., & Lu, Y. (2012, December). ESAMR: An enhanced self-adaptive MapReduce scheduling algorithm. In 2012 IEEE 18th International Conference on Parallel and Distributed Systems (pp. 148-155). IEEE.
19. Zaharia, M., Konwinski, A., Joseph, A. D., Katz, R. H., & Stoica, I. (2008, December). Improving MapReduce performance in heterogeneous environments. In Osdi (Vol. 8, No. 4, p. 7).
20. Chen, Q., Zhang, D., Guo, M., Deng, Q., & Guo, S. (2010, June). Samr: A self-adaptive mapreduce scheduling algorithm in heterogeneous environment. In 2010 10th IEEE International Conference on Computer and Information Technology (pp. 2736-2743). IEEE.
21. Chen, Q., Zhang, D., Guo, M., Deng, Q., & Guo, S. (2010, June). Samr: A self-adaptive mapreduce scheduling algorithm in heterogeneous environment. In 2010 10th IEEE International Conference on Computer and Information Technology (pp. 2736-2743). IEEE.
22. Yang, G. (2011, October). The application of mapreduce in the cloud computing. In 2011 2nd International Symposium on Intelligence Information Processing and Trusted Computing (pp. 154-156). IEEE.
23. Li, Y., Yang, Q., Lai, S., & Li, B. (2015, January). A new speculative execution algorithm based on C4. 5 decision tree for Hadoop. In International Conference of Young Computer Scientists, Engineers and Educators (pp. 284-291). Springer, Berlin, Heidelberg.
24. Javadpour, A., Wang, G., Rezaei, S., & Li, K. C. (2020). Detecting straggler MapReduce tasks in big data processing infrastructure by neural network. The Journal of Supercomputing, 76(9), 6969-6993.
25. Baker, R. (1998). Genetic algorithms in search and optimization. Financial Engineering News, 2(3), 1-3.
26. Kermani, B. G., White, M. W., & Nagle, H. T. (1995, September). Feature extraction by genetic algorithms for neural networks in breast cancer classification. In Proceedings of 17th International Conference of the Engineering in Medicine and Biology Society (Vol. 1, pp. 831-832). IEEE.
27. Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., & Wu, A. Y. (2002). An efficient k-means clustering algorithm: Analysis and implementation. IEEE transactions on pattern analysis and machine intelligence, 24(7), 881-892.

ISSN: 0010-8189