Journal of Harbin Engineering University Vol 45 No. 4
ISSN: 1006-7043 April 2024

Straggler Task Prediction Based Balanced Hadoop for Big Data Using Hladagrad-Enn and Lcv-Choa Based Backup Node Selection

[1]Hetal A. Joshiara , [2]Chirag S. Thaker

[1]Assistant Professor, L.D. College of Engineering, Gujarat Technological University

[2]Professor, L.D. College of Engineering, Gujarat Technological University

Abstract: Data analysis speed is of great significance in Big Data processing. For big data analytics, one of the popular frameworks is Hadoop. Grounded on the MapReduce (MR) model, Hadoop runs applications on a cluster of huge commodities numbers along with less expensive computing nodes since it is a distributed computing framework. Owing to unbalanced load, inaccurate node selection, and unidentified straggler tasks, there remain challenges like high response time, runtime or execution time, and poor handling of resources. The work has developed a Straggler Task Prediction centered Balanced Hadoop (STP-BHADOOP) framework to rectify the existing problem. The computational infrastructure of Big Data's efficiency is enhanced by this framework; thus, accelerating data processing to identify straggler tasks by Speculative Execution (SE). Using the ROS-Flubber (Random Over Sampler based flubber) technique, Hadoop computing is utilized here by distributing a balanced load, totally named Balanced Hadoop (BHadoop). To make a structured data format, extraction of task-centered features and preprocessing are eventuated. To predict the straggler task, the most relevant data is chosen by using the XI-MO technique. Finally, using the HLAdagrad-ENN technique, the prediction is eventuated centered on selected data. The node-centered features are extracted after the straggler task prediction. If a straggler task is identified, nodes selection takes place using the LCV-ChOA technique with the aid of extracted node features. The proposed framework achieves a low runtime, and response time and predicts the straggler task with better precision and recall value, when analogized with prevailing methods is obtained from experimental outcomes.

Keywords: *Big data, Hadoop, MapReduce, Speculative execution, Straggler task, Straggler task prediction based balanced Hadoop (STP-BHADOOP), Random Over Sampler based flubber (ROS-Flubber), Xavier initialization based Mayfly optimization(XI-MO), Hinge loss adaptive gradient-based Elman neural network (HLAdagrad-ENN), and linear coefficient vector based chimp optimization(LCV-ChOA)*

Introduction

Over the last 12 years, the big data analysis platform has accumulated unprecedented amounts of gigantic data processing i.e. far beyond the storage capacity of any one machine. Popularized frameworks were applied by enterprises together with research institutes to handle large data volume that aids to extract applicable data. Apache Spark popularly known as an open-source MR programming platform is a typical example [1]. Spark spotlights the Resilient Distributed Dataset's (RDD) abstract when analogized with Apache Hadoop [3] along with other distributed computing schemes [4]. For efficient performance on high-iterative jobs, it takes an in-memory computing advantage. Straggler is defined as the extension of the stage's execution time by little slow-running tasks of the last wave tasks since more tasks run in corresponding to process elements in RDD partitions [5]. For numerous reasons, stragglers can arise but determining the ultimate one is tough [6]. Straggler is identified by MR systems along with speculative copy restarting on an alternative node termed SE. The last few slow-running tasks are simply backed up by Google. It was also observed that lessen in job execution time by 44% is made by SE [7]. SE is implemented specifically in Spark and others due to improvement in performance [8]. Regarding the map phase or the reduce phase, the SE is configured. For those straggling tasks, the master schedules the speculative tasks when the SE is enabled and arranges them in the queue [9]. When there exist available slots, it will be launched.

Running at most one speculative task at a time is ensured by the scheduler for every original task. If the speculative task ends, the original task is ended and vice versa. As the performance is degraded, SE is disabled by some organizations on some jobs and tasks [10].

HadoopNaive is the original SE mechanism in Hadoop. As task progress is utilized to determine straggler tasks, it shows poor performance in heterogeneous environments. To optimize SE from further aspects, various optimized schemes are designed [17], such as LATE, MCP, ERUL, and so on. Centered on the real-time task's remaining time estimation, straggler tasks are determined in the proposed scheme. But, improper node allocation is led by inaccurate estimation. The cluster's overall performance is affected by the SE performance if multiple "Straggler" occurs in the cluster at the same time. So, it is vital to schedule the backup tasks strategy [18]. The work has proposed an STP-BHADOOP framework centered on SE challenges to tackle the existing issues.

The paper's structure is arranged as follows: the prevailing SE strategies and defects are presented in Section 2. The proposed strategy's detail is signified in Section 3. Section 4 illustrates the experiments and analysis which support the contributions and, finally, the paper gets concluded in Section 5.

2. Literature Survey

Zhongming Fu et al. [19] developed a strategy named ETWR to enhance the efficacy of SE in Spark. To tackle the SE's three key points: straggler identification, backup node selection, and effectiveness guarantee, a heterogeneous environment was measured. Initially, the task was fragmented into sub-phases centered on the task type classification. To find the straggler promptly, it utilized the processing speed together with the progress rate within a phase. Secondly, to estimate the task's execution time, it utilized the Locally Weighted Regression model. Thirdly, to guarantee the speculative tasks' effectiveness, it presented the iMCP model. The experiment showed that when analogized with the Spark-2.2.0, the ETWR could diminish the job execution time by 23.8%, along with enhancing the cluster throughput by 33.2%. But, owing to the unbalanced MR load, execution time was high.

Amir Javadpouret al. [20] developed a backpropagation neural network's application on the Hadoop for the straggler task's detection. It also estimated the tasks balance execution time which was vital in straggler task detection. To detect straggler tasks and achieve accurate estimation of execution time, outcomes attained were analogized with familiar algorithms in the specific domain such as LATE, ESAMR, and the real balance time for Word Count along with Sort benchmarks. The high challenge here was the straggler task node selection.

MandanaFarhanget al. [21] illustrated a dynamic scheme to discover straggler tasks in heterogeneous environments. To estimate task execution's stage weights for accurate estimation of task execution time, a neural network algorithm named SEWANN framework was utilized. To assess the balance execution time and enhance the big data's efficiency, an error was reduced. Both estimated with actual weights were computed along with implementing this method in Hadoop open-source software initially. SEWANN excelled over SVR, Decision Trees, ESAMR, and LATE as baseline methods at 99%, 81%, 85%, and 99%, respectively. Task execution time was enhanced by SEWANN when analogized with baseline method ESAMR by 15%, along with LATE by 24%. Though the work was executed well, diminish in test accuracy occurs due to underfitting led by task characteristic-based straggler prediction.

Laiping Zhao et al. [22] presented Clio, a cross-layer interference-aware optimization scheme that could successfully diminish stragglers for data processing models. Both maps along with reduced tasks scheduling were supported by Clio. In proportion to each worker node's actual computing ability, intermediate data was heuristically dispatched. To consider numerous straggler factors together with balancing the tasks completion times, it was much finely estimated. Clio was implemented in Apache Spark. Grounded on both synthetic and real datasets, its performance was evaluated. When analogized with prevailing models' experimental outcomes, Clio can accelerate the applications' execution by up to 67%. However, high response time and estimated runtime had happened.

Fengjun Shang et al. [23] developed a speculative task scheduling scheme centered on SDN technology. Grounded on ASD, a bandwidth-aware speculative task run-time evaluation strategy was initiated. It accurately speculated the backup task run-time along with providing bandwidth assurance for the speculative task. Finally, simulation experiments were authorized by BWRE. From experimental outcomes, the shortening job turnaround time was outperformed by BWRE at an average of 9.85%. For predicting tasks, it showed a low precision as well as recall.

Haizhou Du et al. [24] systematically conducted the exploration of the elementary issue of automatic, adaptive straggler identification on a big data analytics platform. To solve the complex online optimal issue, Reinforcement Learning (RL) techniques were utilized. To recognize stragglers free from human intervention, employment of Reinforcement learning adaptively opted the optimal parameters. For launching speculative tasks on the heterogeneous cluster, the Hawkeye technique identified stragglers by reinforcement learning at runtime. Hawkeye managed to diminish the job completion time over the different applications type as per experimental outcomes. Centered on a 23% improvement on the current resolutions preciseness to the heterogeneous cluster, an instance revealed a 37% decrease in average job completion. For different scale data, it was highly complex.

3. Proposed Straggler Task Prediction Based On Balanced Hadoop Frameworks

Using the MR framework, distribution along with parallelization of large–scale data processing is carried out. The workload is divided into several MR jobs by this system and gets distributed over multiple nodes. For completing a task, a node's poor performance makes the job consume more time for completion known as Straggler Task. The overhead's major source in parallel programs such as MR is Load imbalance. Tasks with large data become stragglers owing to the input data's uneven distribution. It may delay the overall job completion. As per the task progress, the task's implementation is determined by the SE mechanism. For SE, the load imbalance problem, choosing backup nodes relying on straggler task, node failure remains a great challenge. Using the STP-BHADOOP framework, a prediction-centered balanced SE in a heterogeneous environment is engendered which is illustrated in figure 1.

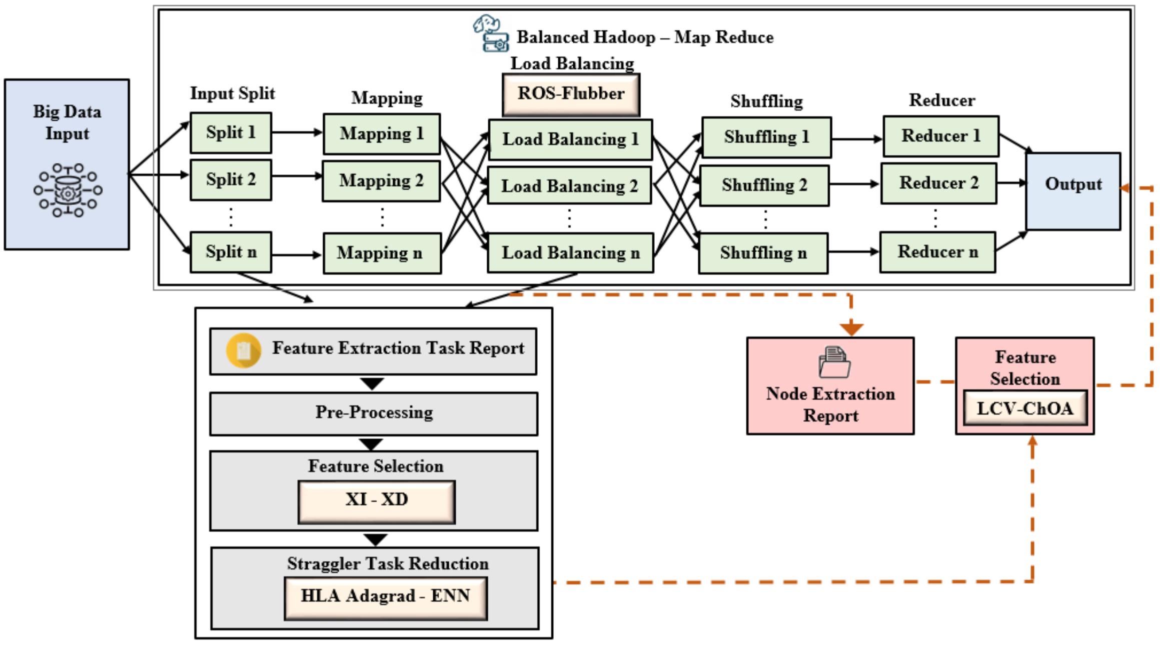

Figure1: Proposed STP-BHADOOP framework

3.1. Big data

Initially, for the work, large datasets collections that cannot be processed using traditional computing techniques are used. For example, the data volume needed by Facebook or YouTube is gathered and managed routinely.

3.1.1. HADOOP

Under Hadoop, data gathered is processed. The data goes through the following phases in Hadoop:

Input phase

Here, fragmenting the texts into fixed-size pieces is named input splits. It is utilized as an input chunk that gets preceded into the mapping process

$$\mathrm{T}_i^n = \left[\mathrm{T}_1^1 \mid \mathrm{T}_2^2 \mid \mathrm{T}_3^3 \mid .... \mid \mathrm{T}_n^n\right] \tag{1}$$

Mapping

Here, to generate output values, the split data is transferred to a mapping function. From input slits, every word's frequency count is performed in the mapping phase.

$$\mathrm{M}_i^n = C_i^n\left[\mathrm{T}_1^1 \mid \mathrm{T}_2^2 \mid \mathrm{T}_3^3 \mid .... \mid \mathrm{T}_n^n\right] \tag{2}$$

Shuffling

From the mapping phase output, relevant records are consolidated by shuffling. The websites frequently obtained are clubbed together.

$$S_i^n = SF_i^n\left[\max\left(C_i^n\right)\right] \tag{3}$$

Reducing

Here, from the shuffling stage, the output values are aggregated. From the shuffling phase, the values are combined in this phase along with frequently occurring websites and form into one. This phase summarizes the complete dataset in short.

$$R_i^n = \mathrm{Re}\,duce\left[S_i^n\right] \tag{4}$$

Centered on key/value pairs, the MR programming matches the Map tasks' output that is transported to diminish the task modes. By a Reduce node, the data with the same key can be processed. For most of the data, the Reduce node task will engender an unbalanced load, if the data matches a particular key or various keys.

3.1.2. Load balancing

For the same quantity of data provided for each task, the same amount of time is required to complete it. No assurance is given for fair data division betwixt jobs or the same quantity of input data processing for a similar time period. For faster runtime, this work engendered a ROS-Flubber to balance the load. Except for the input size along with reducer processing, flubber is similar to the genuine user job. The actual input's parts will suffice to assume the considered sample summarizes the whole input, as the pre-job is mainly utilized for statistical estimation. From the input dataset, randomly picking up a bad proportion of sample records leads to Flubber's poor sampling rate. ROS is used to avoid this. A copy of data sets (B_{data}) is executed in a minor distributed dataset ($\alpha_{\min ority}$). The increase in overfitting's livelihood and the higher sampling rate are provided

$$\Gamma = Randomoversampler\left(\mathrm{B}_{data} + \forall_s\left(\alpha_{\min ority}\right)\right) \tag{5}$$

The user specifies the sample size Γ relative to the input size. Collecting statistics from the mapper's output is the succeeding phase after a run on the sample input. To retrieve the map output size (M_o^n), this is the simplest way. The reducer's optimal number ($\Re$) is formulated as:

$$\Re = \left\lceil \frac{\mathrm{M}_o^n}{\Gamma \times \mathrm{I}_m} \right\rceil \tag{6}$$

Where, the reducer buffer size in Mb is denoted as I_m.

The above phrase's reasoning is given as follows. During the merging phase, the mapper output will be fragmented as the reducer buffer doesn't leak any records in an ideal world. The entire input data's output size is $\frac{1}{\Gamma}$ of the sample map output assuming the mapper's output is proportional to the input's size. The data amount each reducer processes is less than the reducer buffer size I_m is how values are shared. The assumption of all keys having an equal number of values is made finally. A key, whose projected size exceeds the reducer memory buffer size is assigned by the reducer to avoid a straggler reducer node. The reducer's optimal number required is computed to process the remaining keys. In the sample, the list of all the keys is assumed as ζ . Let K be the set of all the keys such that their projected value over the entire dataset is given as:

$$\frac{\mathrm{B}_{Data}^{Size}(k_i)}{\Gamma} > \mathrm{I}_m \tag{7}$$

Where, in the set K , k_i is a key. For finding the reducers' numbers the modified formula is as follows.

$$\Re = |\mathrm{K}| + \left\lceil \frac{\sum_{k_i \in \zeta - \mathrm{K}} \mathrm{B}_{Data}^{Size}(k_i)}{\Gamma \times \mathrm{I}_m} \right\rceil \tag{8}$$

By the above formula, the keys numbers less than the reducers number are considered and computed. This leads to the formulation,

$$O_{\Re} = \min\left(\frac{|\mathrm{K}|}{\Gamma}, \Re\right) \tag{9}$$

Among all reducers after completing the computation, the load is balanced. Generating partition mapping by load-balancing where each key with larger data is allocated to one reducer while the remaining reducer is assigned with smaller keys along with keys not included in the sample. The keys containing larger data quantities that are not jumbled up with smaller keys are ensured.

3.2. Feature extraction for Task

Data are processed under balanced Hadoop along with extracting various task features like competition for local disk, CPU, network bandwidth, memory, to analyze the task for straggler avoidance. Further, owing to faulty hardware along with misconfiguration, the straggler could also be caused. Stragglers are raised by the dynamically altering resource contention outline on an underlying node. Before task launching on a node, for memory, network, disk, CPU, along with other operating system level counters defining the degree of concurrency, the performance counters were collected centered on the findings. Multi broad categories spanned by the counters collected are as follows:

1. CPU utilization $\left(CPU_{\backslash i}^{task}\right)$: CPU idle time $\left(C_{idt}\right)$, system time $\left(s_t\right)$, user time $\left(u_t\right)$ and speed of the CPU $\left(Sp\right)$, etc.

2. Network utilization $\left(Net_{\backslash i}^{task}\right)$: Number of bytes sent and received $\left(b_s \ and \ b_R\right)$, statistics of remote read and write statistics of RPCs $\left(RPC_r \ and \ RPC_w\right)$ etc.

3. Disk utilization $\left(Disk_{\backslash i}^{task}\right)$: The local read and write statistics from the data nodes $\left(dn_r \ and \ dn_w\right)$, amount of free space $\left(F_S\right)$, etc.

4. Memory utilization $\left(Mem_{\backslash i}^{task}\right)$: Amount of virtual $\left(Amt_{\backslash i}^{Vm}\right)$, physical memory available $\left(Pmem_{\backslash i}^{task}\right)$, amount of buffer space $\left(\mathrm{I}_m\right)$, cache space $\left(space_{\backslash i}^{cac}\right)$, shared memory space available $\left(\Theta_{\backslash i}^{ss}\right)$, etc.

5. System-level features $\left(SF_{\backslash i}^{task}\right)$: Number of threads in different states (waiting $\left(w_t\right)$, running $\left(Mem_{\backslash i}^{task}\right)$, terminated $\left(t_t\right)$, blocked $\left(B_t\right)$, etc.), memory statistics at the system level. In total, 107 distinct features are collected that characterize the state of the machine.

6. Remaining time of task $\left(\mathrm{Re}\,m_{\backslash i}^{task}\right)$: (Process speed $\left(\Im_{\backslash i}^{Speed}\right)$ + Remaining data to process $\left(r_{\backslash i}^{task}\right)$ + Sum of the remaining time left in each Phase $\left(\mathrm{Re}\,m_t^t\right)$)

$$\mathrm{Re}\,m_{\backslash i}^{task} = \left(r_{\backslash t}^{task} / Bw_{t}^{task}\right) + \sum Est\left(\mathrm{Re}\,m_{t}^{t}\right) * \hbar$$

(10)

Where,

$$\hbar = \mathrm{M}_{i}^{n}\left(\mathrm{B}_{data}\right) / avg\mathrm{M}_{i}^{n}\left(\mathrm{B}_{data}\right)$$

(11)

7. Remaining time of Shuffle phase $\mathrm{Re}\,m_{t}\left(S_{i}^{n}\right)$ separately (To avoid speed fluctuation)

$$\mathrm{Re}\,m_{t}\left(S_{i}^{n}\right) = \%\mathrm{M}_{i}^{f} - \%S_{i}^{f} / task_{speed}\left(S_{i}^{n}\right)$$

(12)

Where, the finished map task's percentage is denoted as $\%\mathrm{M}_{i}^{f}$ and $\%S_{i}^{f}$ denotes the finished shuffle task's percentage.

From each workload's jobs, multiple tasks may run on every single node. To make a simpler notation, the extracted data are formed into a data frame.

$$\wp_{i} = \left[CPU_{i}^{task} + Net_{i}^{task} + Mem_{i}^{task} + Disk_{i}^{task} + SF_{i}^{task} + \mathrm{Re}\,m_{i}^{task} + \mathrm{Re}\,m_{t}\left(S_{i}^{n}\right) + \ldots \wp_{n}\right]$$

(13)

3.2.1. Preprocessing

Converting the extracted unstructured features into a structured data format is eventuated by preprocessing along with diminishing the error probability. By removing repeated data, noises caused due to outlier, missing values, etc, the data is cleaned. Major preprocessing of data like handling of missing and Nan values, data time variables handling, and data scaling are executed by this work.

3.2.1.1. Handling of missing and Nan values

Nan values signify 'Not a Number' which is a numeric data type member that can be read as a value. It is undefined or unrepresentable while missing values illustrated 'no data present' or 'blank' for a certain feature or group of features. Eliminating the missing and Nan data in the dataset may destroy the valuable data along with diminishes the straggler task prediction's accuracy. It is necessary to deal with the missing and Nan Value to have a successful conclusion.

$$\Psi_{pre} = \aleph_{fxn}\left(\wp_{i\times j}^{Y}\right)$$

(14)

Where, the function handling missing and Nan values are denoted as $\aleph_{fxn}$. Grounded on categorical and numerical values, the methods are selected. Some common methods are Median, Mode, Backward Fill, Forward Fill, Imputation, etc, $\wp_{i\times j}^{Y}$ illustrated the i^{th} row and j^{th} column in the dataset. Without Nan value and missing value, the dataset is obtained and framed into the data frame i.e.

$$\wp_{n}^{DS} = \left[\wp_{1}^{DS}, \wp_{2}^{DS}, \wp_{3}^{DS}, \wp_{4}^{DS}, \ldots.. \wp_{n}^{DS}\right]$$

(15)

3.2.1.2. Handling of date-time variables

For task handling, date and time comprise the informative data. Investigate task trends in space and time, it provides a valuable information source to be used. It studies how Spatio-temporal information can be incorporated to utilize in modeling and forecasting straggler. To turn them into valuable information, the data-time variables require some conversion. Some valuable information is lost by ignoring data and time variables.

$$\Psi_{DT} = \aleph_{Datetime}\left(\wp_{n}^{DS}\right)$$

(16)

Where, the function that handles the date and time variables is represented as $\aleph_{Datetime}$.

3.2.1.3. Scaling

To obtain the same units dataset feature that normalizes the independent variables range or data features, features scaling is done. To scale down the features, the Minmax Scaler is utilized within a similar range. It calculates the feature vector betwixt the range of 0 and 1. The Minmax scaler is formulated as:

$$\Psi_{Sca} = \frac{\wp_{i\times j} - \min\left(\wp_{i\times j}\right)}{\max\left(\wp_{i\times j}\right) - \min\left(\wp_{i\times j}\right)}$$

(17)

Thus, to obtain healthier data for diminishing error rate, the data is being preprocessed.

3.2.2. Backup Node selection

To avoid the curse of dimensionality, feature selection chooses the most relevant features from the extracted features. It leads to informative data loss for predicting straggler tasks. Centered on mayfly social behavior, the most vital trait is chosen by the feature selection model, notably their mating process. After emerging from the egg, mayflies are thought to be adults. The fittest mayflies stay alive despite the duration they live. The problem's solution is represented by every mayfly in the search space. The current Mayfly optimization becomes caught in the local optimal solution during the positive attraction constant value. While updating the male and female mayflies' positions, the search speed gradually slows down. The work has developed Xavier initialization-based Mayfly optimization (XI-MO) to avoid the trap down. It maintains the fly population's variance and defines the minimum and maximum value to maintain the balance between local optimum and global optimum. To find out the coefficient vector β_1 and β_2 , Xavier initialization is used that gives a balanced exploration as well as exploitation rate aiding to attain a faster convergence rate. The selection process is speeded by the developed FS technique that performs accurately with less error probability. The developed algorithm works as follows;

The male and female mayfly populations (extracted features) are generated randomly initially. The dataset is also split as 50 % for males and 50 % for females centered on the flies. For each problem in the population search space, each fly (best feature) is represented as a candidate solution that is a d-dimensional vector $\wp = \wp_1, \wp_2, \wp_3 ...., \wp_n$, and the performance is estimated on objective function ($Obj(\wp_i)$).

For each mayfly, the position change along with the flying direction for the individual as well as social flying experience is given by the velocity $v = v_1, v_2, v_3 ...., v_n$. Particularly toward its personal best position ($\wp_i^p$), each mayfly adjusts its trajectory. The best position attained by any swarm's mayfly is ($\wp_i^g$).

Centered on its own experience, the male mayfly movement is evaluated as that of its neighbors. The mayfly's current position i in the search space at time step t is represented as $\wp_i^t$. The position is modified to the current position by adding a velocity v_i^{t+1} as,

$$\wp_i^{t+1} = \wp_i^t + v_i^{t+1} \tag{18}$$

With $\wp_i^0 \sim U(\wp_{\min}, \wp_{\max})$

The male mayflies's velocity gets down under the nuple dance scenario. Finally, the male mayfly's velocity i is calculated as:

$$v_{ij}^{t+1} = v_{ij}^t + \beta_1 e^{-\alpha r_P^2}\left(\lambda_{ij}^p - \wp_{ij}^t\right) + \beta_2 e^{-\alpha r_P^2}\left(\lambda_{ij}^g - \wp_{ij}^t\right) \tag{19}$$

$$\beta_1 = -\sqrt{\frac{6}{\wp_i^t + \wp_i^{t+1}}} \tag{20}$$

$$\beta_2 = \sqrt{\frac{6}{\wp_i^t + \wp_i^{t+1}}} \tag{21}$$

Where, the mayfly's velocities in dimension at a time step is modeled as v_{ij}^t, the mayfly's positions i in dimension j at a time step t is interpreted as $\wp_{ij}^t$, β_1 and β_2 are positive attraction coefficient vectors that are used to scale the contribution of the cognitive and social components respectively. Furthermore, the mayfly's best position i had ever visited is λ_i^p. The personal best position λ_{ij}^p at the subsequent time step $t+1$, is computed considering minimization problems as:

$$\lambda_i^p = \begin{cases} \wp_i^{t+1}, & if\ f\left(\wp_i^{t+1}\right) < f\left(\aleph_i^p\right) \\ is\ kept the\ same, & otherwise \end{cases} \tag{22}$$

Where, the objective function $f : Rn \rightarrow R$ estimates the solution's quality. At time step *t*, the global best position $gbest$ is defined as,

$$\lambda_i^g \in \left\{\lambda_1^p, \lambda_2^p, \lambda_3^p, \ldots. \lambda_n^p \mid f\left(best^c\right)\right\} = \min\left\{f\left(\lambda_1^p\right), f\left(\lambda_2^p\right), f\left(\lambda_3^p\right), \ldots\ldots, f\left(\lambda_n^p\right)\right\}$$

(23)

Where, the male mayflies' total number in the swarm is denoted as n.

Finally, in eq. (19) a fixed visibility coefficient α is utilized to limit a mayfly's visibility to others, while the Cartesian distance between c_i and λ_i^p is denoted as r_p and the Cartesian distance between c_i and λ_i^g is notated as r_G. These distances are calculated as

$$\left\|\wp_i - \wp_i^n\right\| = \sqrt{\sum\left(\wp_{ij} - \wp_{ij}^n\right)^2}$$

(24)

Where, the j^{th} element of mayfly i is notated as $\wp_{ij}$ and $\wp_i$ corresponds to $\wp_i^p$ or $\wp_i^g$.

To function the model properly, the swarm's best mayflies execute their up-and-down nuptial dance. Finally, changing their velocities constantly is made by the best mayflies which are calculated as,

$$v_{ij}^{t+1} = v_{ij}^t + D.R$$

(25)

Where, the nuptial dance coefficient is denoted by D and R along with [-1, 1] random value range. A stochastic element is introduced by this up and down movement to the algorithm. As the female flies don't gather instead fly approach males to breed, unlike the male flies. Hence, the female mayflies' movement is computed. The female mayflies' i current position in the $y$search space is assumed as x_i^t at a time step t. By summing velocity γ_i^{t+1} to the i current position, the position is changed, i.e.

$$y_i^{t+1} = y_i^t + v_i^{t+1}$$

(26)

With $y_i^0 \sim U\left(y_{\min}, y_{\max}\right)$

Towards the best male, the best female flies get attracted. Taking into account the minimization problems, consequently, the velocities are computed as:

$$v_i^{t+1} = \begin{cases} v_{ij}^t + \beta_2 e^{-\alpha r_{mf}^2}\left(\wp_{ij}^t - y_{ij}^t\right) & if\ f(y_i) > f(\wp_i) \\ v_i^{t+1} + fl * R & if\ f(y_i) \le f(\wp_i) \end{cases}$$

(27)

Where, the female mayfly's velocity i in dimension $j = 1,2,3,4\ldots,n$ at a time step t is modeled as v_{ij}^t, the female mayfly's position i in dimension j at a time step t is depicted as v_{ij}^t, the positive attraction coefficient vector and fixed visibility coefficient are denoted as β_2 and α, while the Cartesian distance betwixt male and female mayflies is signified as r_{mf}. Finally, when a female is not fascinated by a male, fl is a random walk coefficient utilized. Thus, it flies at random and R is a random value in the range [- 1, 1].

After, the fly takeovers mating that is crossover operator: from the male population together with the female population, one parent is chosen. Similarity occurs in the process of choosing parents and the process of females being attracted to the males. Either randomly or centered on their fitness function, the selection is eventuated. Later, the best male is bred by the best female; the second-best male is bred by the second-best female, and so on. Two offspring are produced by the crossover as follows,

$$\Phi_1 = \wp * f(\wp_i) + (1-\wp) * f(y_i)$$

(28)

$$\Phi_2 = \wp * f(y_i) + (1-\wp) * f(\wp_i)$$

(29)

Where, the male parent is proffered as $f(\wp_i)$, the female parent is notated as $f(y_i)$ and a random value within a specific range is illustrated as $\wp$. Zero is set as the initial velocity for Offspring. Replacing the worst solution with the best ones, pbest and gbest values are updated.

$$FS_1^w = \left[\wp_1^w, \wp_2^w, \wp_3^w, \wp_4^w, \wp_5^w \wp_6^w, \wp_7^w \ldots \wp_n^w\right]$$

(30)

3.3. STRAGGLER TASK PREDICTION

Under the proposed HLAdagrad –ENN, the selected features are trained to identify the task as a straggler or not. Due to overfitting and underfitting, the prevailing ENN methods led to a high error rate. It also consumes more time to diminish the error that leads to inaccurate straggler tasks identification. Figure 2 shows the HLAdagrad optimizer used with ENN to solve the issue. The proposed work undergoes data training inside imperative layers, say input layer (IL), Hidden Layer (HL), undertake layer together with Output Layer (OL) before following those conditions.

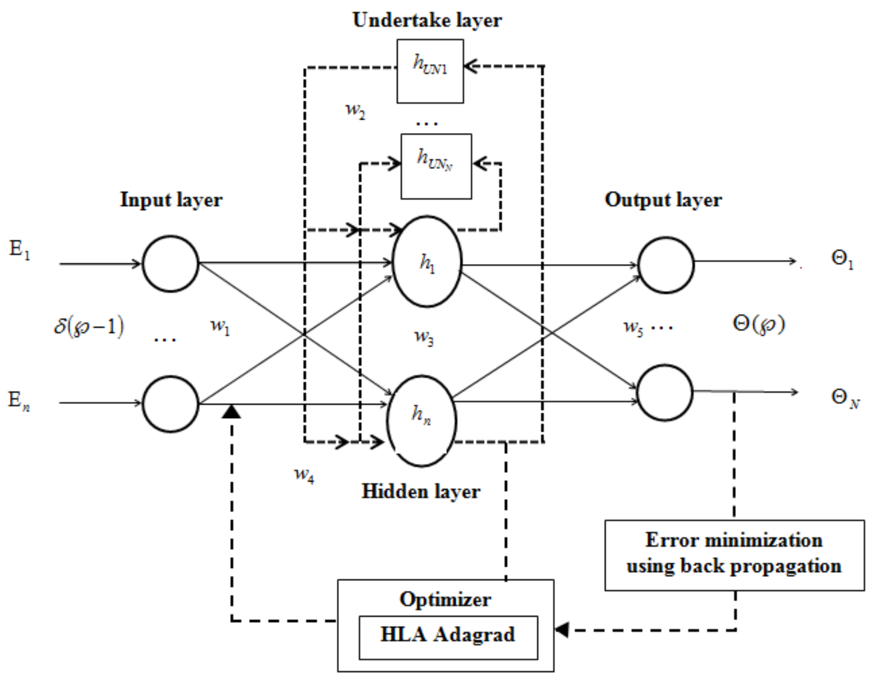

Figure2: Proposed HLAdagrad –ENN architecture

Suppose with the input E_n , h_n is the number of HL , h_{UN} is the undertake neurons, w_1 is the weight of IL to HL, w_2 , w_3 , w_4 is the weight of undertake layer to HL , the weight of HL to OL is w_5 , the neural network's input is proffered as $\delta(\wp-1)$, the output of HL is depicted as $h_n(\wp)$, the output of undertake layer is denoted as $h_{UN}(\wp)$, and the output of neural network is proffered as $\Theta(\wp)$; then,

$$h_n(\wp) = \mu\big(w_4 h_{UN}(\wp) + w_3 h_{UN}(\wp) + w_2 h_{UN}(\wp) + w_1(\delta(\wp-1))\big)$$
(31)

Where,

$$h_{UN}(\wp) = h(\wp-1))$$
(32)

Where, the hidden layer transfer feature is signified as μ , utilized largely in the S-type function, i.e.

$$\mu(\wp) = (1+e^{-\wp})^{-1}$$
(33)

η signifies the OL transfer function, which is often a linear function, that is,

$$\Theta(\wp) = \eta(w_5 h(\wp))$$
(34)

To revise weights, BP is utilized by Elman NN. Assessing the weights is centered on the entropy that is the network's error is rendered by,

$$e = \sum_{K=1}^{m} (\hat{\Gamma}(\Theta) - Y(\Theta))^2$$
(35)

Wherein, for detecting the user the output vector is signified as $\hat{\Gamma}(\Theta)$. BP is not required when the error value is zero; however, using the Hinge loss Adagrad optimizer (HLAdagrad), the BP is formulated for an error value. With low computation time, the optimizer updates the weight and minimizes the loss. To avoid overfitting of data, the hinge loss with Adagrad optimizer helps and data underfitting is not allowed. For frequently occurring features' parameters, smaller updates are performed. For infrequently occurring features' parameters, larger updates are executed. The HLAdagrad optimizer is given by:

$$w_{t+1,i} = w_{t,i} - \frac{\ell}{\sqrt{Z_{t,ii} + \varepsilon}} \frac{\partial w}{\partial e} + \sum_{i=1}^{k} \kappa e_i \tag{36}$$

Where, the hinge loss with κ as constant is denoted as $\sum_{i=1}^{k} \kappa e_i$ and the sum of the overall loss is denoted as e_i, the update weight at iteration i denoted as $w_{t+1,i}$, the current weight is illustrated $w_{t,i}$, the learning rate is represented ℓ, regarding weight, the sum of the gradients of the past gradients is denoted as $Z_{t,ii}$, the constant that is chosen very small i.e.0.001is depicted as ε and the weight's partial derivative is denoted as $\frac{\partial w}{\partial e}$ concerning loss.

As a result, the highest priority is identified for the task having the longest remaining time i.e. SE (Back-Up). In figure 3, the straggler task prediction's outline is illustrated in the form of pseudo-code.

```
Input: Extracted features based on task $FS_1^{w} = [\wp_1^{w}, \wp_2^{w}, \wp_3^{w}, \wp_4^{w}, \wp_5^{w}\wp_6^{w}, \wp_7^{w} .. \wp_n^{w}]$
Output: Output straggler task or not

Begin
    Initialize the weights ($w_1\, w_2, w_3, w_4, w_5$) hidden layers ($h_N$), undertake layers $h_{UN}$
    For n=1 to r
        Evaluate the hidden layer output,
            Evaluate hidden layer transfer feature using,
                $\mu(\wp) = (1 + e^{-\wp})^{-1}$
            Evaluate the hidden layer output using,
                $h_n(\wp) = \mu(w_4 h_{UN}(\wp) + w_3 h_{UN}(\wp) + w_2 h_{UN}(\wp) + w_1(\delta(\wp - 1)))$
            Evaluate the output layer using,
                $\Theta(\wp) = \eta(w_5 h(\wp))$
            Compute error function,
                $e = \sum_{\kappa=1}^{m} (\hat{\Gamma}(\Theta) - Y(\Theta))^2$
            If ($e =\neq 0$)
                Perform BP by updating weights using optimizer,
                    $w_{t+1,i} = w_{t,i} - \frac{\ell}{\sqrt{Z_{t,ii} + \varepsilon}} \frac{\partial w}{\partial e} + \sum_{i=1}^{k} \kappa e_i$
            Else
                Detection of straggler task
            End if
    End for
End Begin
```

Figure3: Pseudo code for HLAdagrad-ENN for STP

3.4

. NODE FEATURE EXTRACTION REPORT

Grounded on the mp- reduce nodes, the feature extraction report gathers the information that aids to choose the backup node for the respective straggler task. The features extracted such as

1. Job success rate: $\left(J_{SR} = PROGRESS_I / \Delta t\right)$

2. Jobs' Priority and the resource's default priority are selected to execute the job: $\left(\Pr i_J(\hat{\Theta}) = \dfrac{Slowtask_i}{\Delta t}\right)$

3. Choosing resource's capacity and the resources available to execute high priority jobs: $\left(\operatorname{Re} s_J(\hat{\Theta}) = taskmem_i - alloted \operatorname{Re} s_i^{mem}\right)$

4. Progress rate centered map task: $\left(\Pr ogress_{map/task} = C_{read} / \mathrm{M}_{IN}\right)$

5. Progress rate between reduce and task: $\left(\Pr ogress_{reduce/task} = 1/3 \times \left(\Xi_{stage} + C_{read} / R_{IN}\right)\right)$

6. Average progress rate: $\left(\Pr ogress_{AVG} = \sum_{j=1}^{num} PROGRESS_j / N_{Runningtaks}\right)$

7. Left time: $\left(lt_i = \left(1 - PROGRESS_i\right) / \Pr ogressRate_i\right)$

To choose the best backup node for the straggler task, many more features are extracted. For further proceedings, the features are framed into a data frame.

$$N_i = \begin{bmatrix} J_{SR} + \Pr i_J + \operatorname{Re} s_J + \Pr ogress_{map/task} + \\ \Pr ogress_{reduce/task} + \Pr ogress_{AVG} + lt_i + ...N_n \end{bmatrix} \quad (37)$$

3.4.1. Node selection

For the jobs' execution or straggler tasks, the appropriate node selection issue is addressed by the developed node selection algorithm. Grounded on the individual intelligence's social behavior and Chimps' sexual motivation during their hunting process, a linear coefficient vector-centered Chimp Optimization (LCV-ChOA) Algorithm is developed here. The coefficient vector's updation leads to incorrect model driving along with prey chasing in the prevailing ChOA. Due to this, irrelevant selection of the exact resource for the tasks' execution is done. To solve that, LCV is utilized in updating the vectors. The node selection is held grounded on the tasks' priority. The node is selected, when a high-priority task is available. Along with, the task failing chances are diminished. Centered on the straggler prediction model, each task's priority is assigned.

Initially, the chimps population (i.e., the extracted features N_i) that live in a fission-fusion society is classified as,

a) Drivers,

b) Barrier,

c) Chaser, and

d) Attackers (best solutions).

The exploration stage and the exploitation stage are the two stages in the chimps' hunting process. Driving, blocking, and chasing are included in the exploration stage whereas the exploitation stage comprises attacking the prey (nodes).

In driving, the driver does not attempt to catch the prey but rather only to follow it. In blocking, the chimps positioned themselves in trees along with prey's gateway route is hindered. In chasing, running after the prey to catch is made by chasers. For attacking the prey, the attackers predict the optimal route in exploitation.

The prey's driving and chasing can be mathematically modelled as,

$$D = | C.N_p(t) - \upsilon.N_c(t) | \quad (38)$$

$$N_c(t+1) = N_p(t) - A.D \quad (39)$$

Where, the coefficient vectors are mentioned as A and C, with respect to various chaotic maps the chaotic vector is computed as υ, the position of the prey and a chimp at the number of current iterations q are given as $N_p(t)$ and $N_c(t)$. During the hunting process, the sexual motivation of the chimps' effect is expressed by the chaotic vector υ. Using LCV, the coefficient vectors are calculated as,

$$A = \frac{1}{4}\log\left[i + \frac{1}{i_{\max}}\right]\phi \tag{40}$$

$$C = 2\Omega \tag{41}$$

Where, using robust confidence intervals $Con.Int(\phi,\Omega)$, $\phi\, and\ \Omega$ are the parameters updated to balance the exploration and exploitation rate. The robust confidence intervals are calculated as,

$$Con.Int(\phi,\Omega) = med \pm \varepsilon_{(\frac{\gamma}{2},n-1)} \frac{\sigma}{\sqrt{N_i}} \tag{42}$$

Where, the confidence coefficient is notated as γ , the confidence interval's percentage point is signified as ε , the sample standard deviation is expressed as σ , and the sample median is denoted as med . Centered on the calculation time along with the system's accuracy, fitness is evaluated.

The attackers find the prey's position with the aid of a driver, barrier, and chasers for attacking the prey. Centered on chaotic strategy, the prey is attacked by the chimps. Exploring the prey's location and encircling the prey are the two approaches employed here. The attacker, driver, barrier, and chaser update the prey's position since the prey's initial position is unknown. The position updation can be expressed as,

$$R_{attacker} = | v_1 * N_A - C_1 * N | \tag{43}$$

$$R_{chaser} = | v_2 * N_{Ch} - C_2 * N | \tag{44}$$

$$R_{barrier} = | v_3 * N_B - C_3 * N | \tag{45}$$

$$R_{driver} = | v_4 * N_D - C_4 * N | \tag{46}$$

To update their solutions, four optimal resolutions are stored along with other chimps. The chimp's next position can be at any current position middle and prey's position, if the random vectors of v lies in [1,-1]. The chimp's location is updated as,

$$N(1) = N_A - A_1.R_{attacker} \tag{47}$$

$$N(2) = N_{Ch} - A_2.R_{chaser} \tag{48}$$

$$N(3) = N_B - A_3.R_{barrier} \tag{49}$$

$$N(4) = N_D - A_4.R_{driver} \tag{50}$$

The chimp's position can be updated from the obtained location as,

$$N(t+1) = \frac{N(1) + N(2) + N(3) + N(4)}{4} \tag{51}$$

By the social benefits like support and sex, the chimps' hunting process can be affected. To forget their involvement in the hunting process, chimps get motivated by this. In the prey attacking final stage, chaotic maps are utilized that aids chimps in diminishing the local optima together with slow convergence rate issues. The parameter $\eta \in (0,1)$ determined the probability of selecting the normal position updation along with chaotic map centered position updation as,

$$N_c(t+1) = \begin{cases} N_p(t+1) - A.R & if\,(\eta < 0.5) \\ \varsigma_{m(pos)} & if\,(\eta > 0.5) \end{cases} \tag{52}$$

Where, the chaotic map-centered position updation process is proffered as ς_m . The attackers attack the prey grounded on an updated position. When the prey's movement stops, the hunting process also gets stopped. The best nodes for the straggler task are obtained finally.

4. RESULTS AND DISCUSSION

In a heterogeneous Hadoop environment, this section validates the proposed STP-BHADOOP framework. It refers to a Hadoop cluster where each node's hardware resources are diverse. These hardware resources comprise a disk, memory, CPU, and so forth. For creating a heterogeneous dispersed Hadoop environment, the four nodes were virtual out with dissimilar hardware resources using VMware virtual machine. Grounded on Runtime, response time, data locality, estimated runtime in the mapping phase, estimated runtime in reduced phase, precision, and recall, the validation is carried out. Clio, Hawkeye, Spark-Speculative, SE Algorithm centered on Decision Tree (SECDT) and longest approximation time to end (LATE) are some prevailing models analogized with the proposed work's attained outcome. With a 128 MB block size HDFS distributed file system, all tests are centered on three blocks. For each Map and Reduce task, two slaves are considered in this work. Grounded on the t-word counting program, the results are generated.

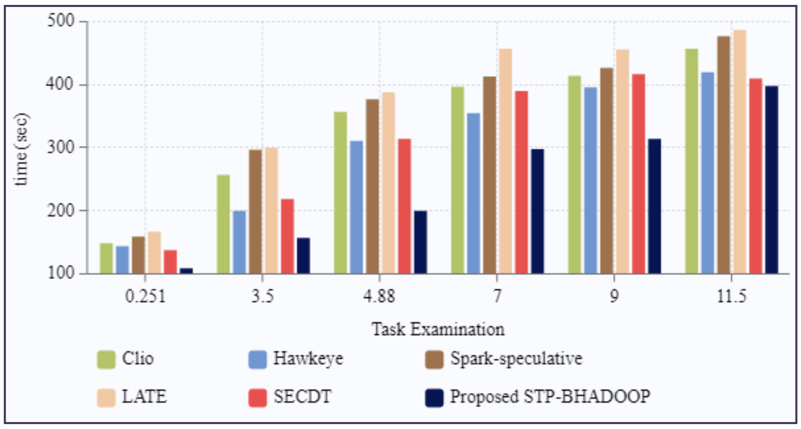

Figure 4: Demonstration of Runtime with two slaves

The proposed methods' runtime is validated in figure 4. The total time taken by the Map and reduce to execute a task is named Runtime. High efficiency was normally attained for low runtime models. Grounded on the various task of various GB ranging from 0.251 to 11.5, the runtime analysis is eventuated. A runtime of 110s for 0.251G and 399s for 11.5G is attained by the proposed framework whereas a runtime ranging between 139s-168s for 0.251G and 411s-488s for 11.5G is shown by the proposed schemes. When analogized with the existing models, less runtime is attained by the proposed framework. When analogized with the prevailing models, faster-balanced task distribution is eventuated in the proposed work

.

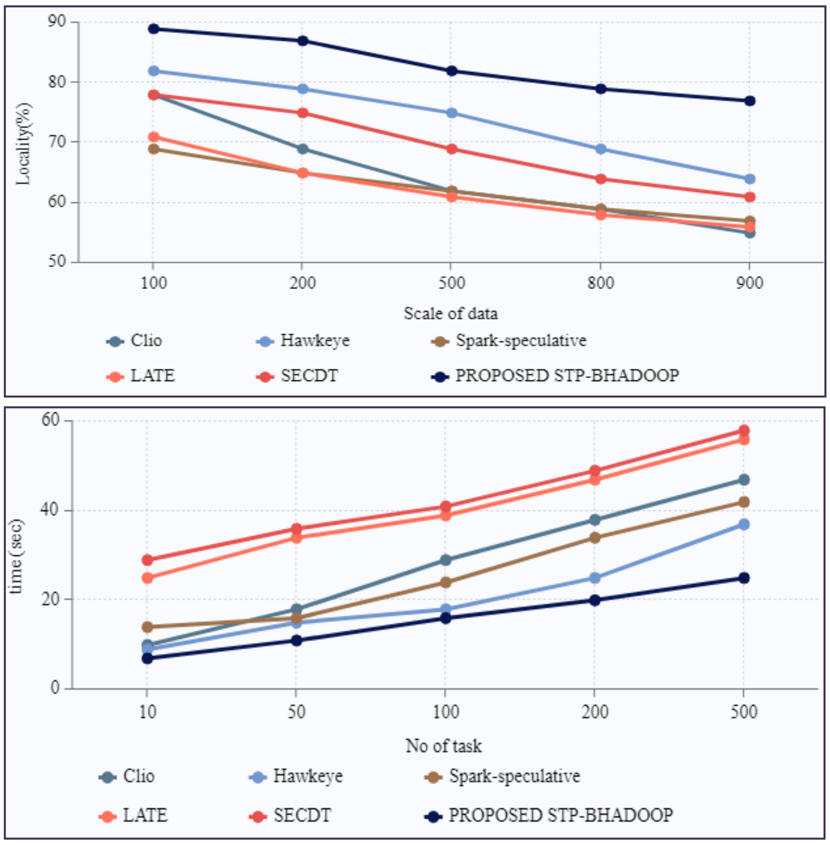

Figure 5: Graphical representation of proposed method based on (a) data locality (b) Response time

The proposed framework's data locality measures are described in figure 5(a). Data locality refers to the fact that the task's executing mode and the task's data blocks storing node are similar nodes. The time taken to transfer data blocks to the execution node is diminished by data locality. Finally, the scheduling algorithm's reasonableness is indicated by the proportion of data locality jobs. A scalable model is indicated by upholding a high percentage of data localities i.e. scalable to different data scales. A data locality of 89% for 100 scales of data and there is a decrease of 12% for the 900 scales of data is attained by the proposed work. However, the prevailing models normally obtain a low data locality for 100 scales of data i.e. ranging between 69%-82% and a range of 55% -61% for 900 scales of data. Better data locality is maintained by the proposed work since a huge difference occurs between the proposed work and the existing work.

The proposed framework's response time is represented in figure 5(b). The time taken by the task to respond from a node is termed Response time. To eliminate data congestion, response time should be low. The proposed method's response time diminishes as the tasks' number increases. i.e. for the 10 tasks, the response time was 7s but as the task number increases the time also increases with a light margin of 25s as shown in the figure. For the overall task, a high RT ranging between 9s-58s is achieved by the prevailing methods that lead to data traffic between nodes.

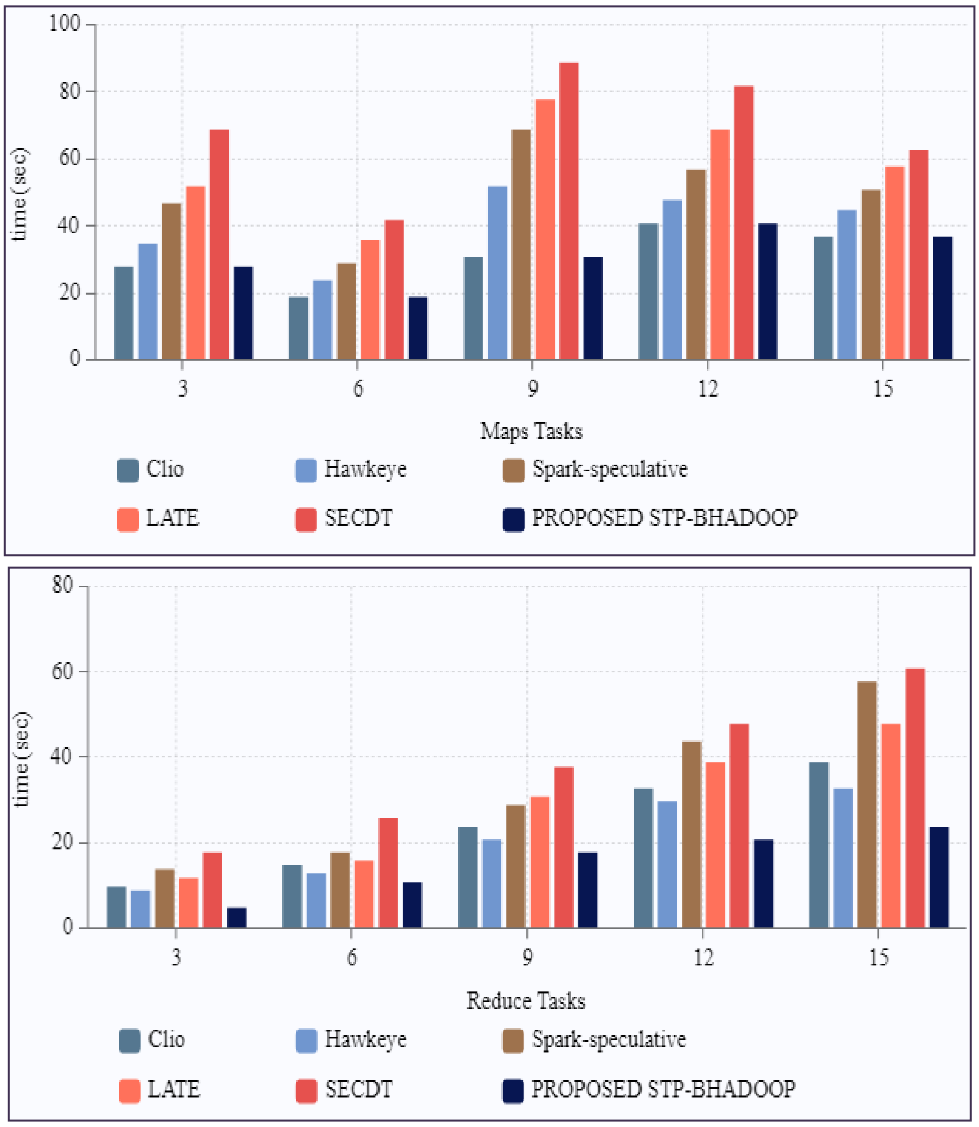

Figure 6: Difference in estimated runtime in (a) Map phase (b) Reduce Phase

Figures 6(a) and 6(b) depict the MR phase's predicted runtime. Importantly, a huge impact on the job's runtime is made by each node's condition. Every run may account for a variable time amount if the working of the node is held as a background task. Grounded on the input data quantity along with the kind of application for further assessment, different runtimes are kept in the database. The projected runtime should be error-free for the previous task. A better MR runtime estimation with a 5% error rate is attained by the proposed method. That is, a Map runtime of 28s for 15 tasks and a Reduce runtime of 24s are attained. Inaccurate SE has resulted from the attainment of a wide error rate of about 30-35 percent by the prevailing methods.

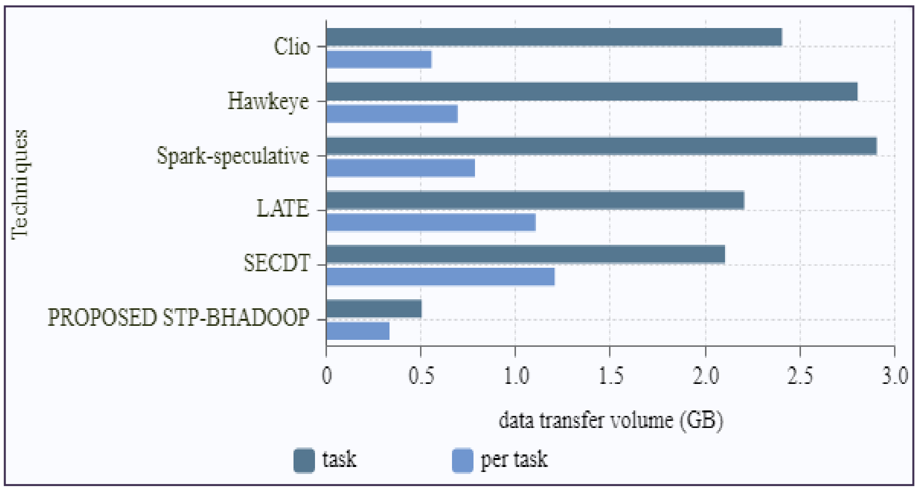

Figure 7: Inner data congestion of backup task

The backup data traffic in the interim state is illustrated in figure 7. Throughout job execution, intermediate data across nodes is retrieved from the result. The volume is counted by the network management software's monitor. From nearby backup nodes, the data may be read by the proposed framework's backup jobs, lower communication time cost reduces backup task time, and finally diminishes stage execution time. Fast nodes and backup task efficacy are ignored by the current technique.

Table1: Statistic for runtime based on precision and recall

Strategy/ metrics	**Precision (%)**	**Recall (%)**	**Average Find Time(sec)**
Clio	65	67.84	98
Hawkeye	78	79.99	87
Spark-Speculative	68	69.78	89
LATE	61	61.98	95
SECDT	63	65.45	102
Proposed STP-BHADOOP	85	86.66	43

Using precision, recall, and average find time, the stragglers' occurrences in each approach are calculated along with runtime statistics verified for suggested frameworks. Table 1 signifies the investigation findings. For locating the straggler, the proposed approach is more accurate than the existing models. The STP-BHADOOP improves accuracy and recall to over 85 % and 86.66 %, respectively with an average search time of 43 sec. This occurs owing to the optimization methods speculative strategy's set, which includes precise straggler identification, optimal node selection, and efficacy assurance. A range of 61 % to 79.99 % for low precision and recall is achieved by the prevailing approaches along with a long average finding time.

5. CONCLUSION

A huge impact on Hadoop clusters is made by MR jobs scheduling and resource allocation in heterogeneous environments. For heterogeneous jobs and resources, complexity occurs in Big data analytics as well as workload management. The SE strategy is made challenging by the tasks characteristics and the runtime environments complexity. The work has developed the STP-

BHADOOP framework to offset this issue. For straggler task prediction, an HLAdagrad-ENNand LCV-ChOA for backup node selection was utilized in this framework. Straggler identification, backup node selection, along with effectiveness guarantee is provided by this work. A balanced load between Mapper and Reducer is provided in this work. High runtime and traffic congestion between the nodes are avoided. The task characteristic's uncertain change is also handled by the proposed framework. Overall, an average data locality of 82.8%, a response time of 25s for performing 500 tasks, and a runtime of 399s for the maximum data size were attained here. It also achieves a precision of 85%, recall of 86.66% for predicting the straggler task as analogized to the state-of-the-art methods.

Reference

1. Maotong Xu, Sultan Alamro, Tian Lan and Suresh Subramaniam, "Chronos a unifying optimization framework for speculative execution of deadline-critical mapreduce jobs", 38th International Conference on Distributed Computing Systems, 2-6 July Vienna, Austria, 2018.
2. Zhongming Fu and Zhuo Tang, "Optimizing speculative execution in spark heterogeneous environments", IEEE Transactions on Could Computing, 2019, **Doi:** 10.1109/TCC.2019.2947674.
3. Kamalakant Laxman Bawankule, Rupesh Kumar Dewang and Anil Kumar Singh, "Historical data based approach for straggler avoidance in a heterogeneous hadoop cluster", Journal of Ambient Intelligence and Humanized Computing, vol. 12, no. 3, pp. 1-17, 2021.
4. Fengjun Shang, Xuanling Chen, Chenyun Yan, Luzhong Li and Yuting Zhao, "The bandwidth-aware backup task scheduling strategy using sdn in hadoop", Cluster Computing, vol. 22, no. 1, pp. 1-11, 2019.
5. Khushboo Kalia and Neeraj Gupta, "Analysis of hadoop mapreduce scheduling in heterogeneous environment", Ain Shams Engineering Journal, 2020, Doi: 10.1016/j.asej.2020.06.009
6. Xue Ouyang, Peter Garraghan, Bernhard Primas, David Mckee, Paul Townend and Jie Xu, "Adaptive speculation for efficient internetware application execution in clouds, ACM Transactions on Internet Technology, vol. 18, no. 2, pp. 1-22, 2018.
7. Neeraja J Yadwadkar, Bharath Hariharan, Joseph E Gonzalez and Randy Katz, "Multi-task learning for straggler avoiding predictive job scheduling", Journal of Machine Learning Research, vol. 17, pp. 1-37, 2016.
8. Ibrahim Adel Ibrahim and Mostafa Bassiouni, "Improving mapreduce performance with progress and feedback based speculative execution", International Conference on Smart Cloud, 3-5 November, New York, NY, USA, 2017.
9. Xiaodong Liu and Qi Liu, "An optimized speculative execution strategy based on local data prediction in a heterogeneous hadoop environment", International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), 21-24 July, Guangzhou, China, 2017.
10. Chunlin Li, Mingyang Song, Qingchuan Zhang and YoulongLuo, "Cluster load based content distribution and speculative execution for geographically distributed cloud environment", Computer Networks, vol. 186, pp-1-22, 2021.
11. Haizhou Du, Shaohua Zhang, Ping Han, Keke Zhang and Bin Xu, "Cheetah a dynamic performance optimization approach on heterogeneous big data analytics cluster", 5th International Conference on Big Data Computing and Communications (BIGCOM), 9-11 August, Qing Dao, China, 2019.
12. Shyam Deshmukh, Komati Thirupathi Rao and Mohammad Shabaz, "Collaborative learning based straggler prevention in large-scale distributed computing framework", Security and Communication Networks, 2021, Doi: 10.1155/2021/8340925.
13. Reshma S Gaykar, Nalini C and Joshi S. D, "Identification of straggler node in distributed environment using soft computing algorithms", International Conference on Emerging Smart Computing and Informatics (ESCI), 5-7 March, Pune, India, 2021.
14. Neda Maleki, Hamid Reza Faragardi, Amir Masoud Rahmani, Mauro Conti and Jay Lofstead, "Tmar a two-stage mapreduce scheduler

for heterogeneous environments", Human-centric Computing and Information Sciences, vol. 10, no. 42, pp. 1-26, 2020.

15. Abolfazl Gandomi, Ali Movaghar, Midia Reshadi and Ahmad Khademzadeh, "Designing a mapreduce performance model in distributed heterogeneous platforms based on benchmarking approach", The Journal of Supercomputing, vol. 76, no. 12, pp. 1-27, 2020.
16. Qi Liu, Weidong Cai, Jian Shen, Zhangjie Fu, Xiaodong Liu and Nigel Linge, "A speculative execution strategy based on node classification and hierarchy index mechanism for heterogeneous hadoop systems", ICACT Transactions on Advanced Communications Technology (TACT), vol. 5, no. 4, pp. 889-894, 2016.
17. Rathinaraja Jeyaraj, Ananthanarayana V. S and Anand Pau, "Improving map reduce scheduler for heterogeneous workloads in a heterogeneous environment", Concurrency and Computation Practice and Experience, vol. 32, no. 9, pp. 1-10, 2019.
18. Ihsan Ullah, Muhammad Sajjad Khan, Muhammad Amir, Junsu Kim and Su Min Kim, "lstpd least slack time-based preemptive deadline constraint scheduler for hadoop clusters", IEEE Access, vol. 8, pp. 111751-111762, 2017.
19. Zhongming Fu and Zhuo Tang, "Optimizing speculative execution in spark heterogeneous environments", IEEE Transactions on Could Computing, vol. 99, pp. 1-15, 2019.
20. Amir Javadpour, Guojun Wang, Samira Rezaei and Kuan-Ching Li, "Detecting straggler mapreduce tasks in big data processing infrastructure by neural network", The Journal of Supercomputing", vol. 76, pp. 6969-6993, 2020.
21. Mandana Farhang and Faramarz Safi-Esfahani, "Recognizing mapreduce straggler tasks in big data infrastructures using artificial neural networks", Journal of Grid Computing, vol. 18, no. 3, pp. 1-23, 2020.
22. Laiping Zhao, Yiming Li, Francoise Fogelman-Soulie and Keqiu Li, "A holistic cross-layer optimization approach for mitigating stragglers in in-memory data processing", Journal of Systems Architecture, vol. 111, pp. 1-12, 2020.
23. Fengjun Shang, Xuanling Chen, Chenyun Yan, uzhong Li and Yuting Zhao, "The bandwidth-aware backup task scheduling strategy using SDN in hadoop", Cluster Computing, vol. 22, no. 1, pp. 1-11, 2019.
24. Haizhou Du and Shaohua Zhang, "Hawkeye adaptive straggler identification on heterogeneous spark cluster with reinforcement learning", IEEE Access, vol. 8, pp. 57822-57832, 2020.