JOURNAL OF CRITICAL REVIEWS

ISSN- 2394-5125 VOL 7, ISSUE 15, 2020

PERFORMANCE OPTIMIZATION IN BIG DATA PROCESSING USING ENHANCED SPECULATIVE APPROACH

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

[1]Research Scholar, Assistant Professor, Department of Computer Engineering, L.D. College of Engineering, Ahmedabad, Gujarat Technological University, Chandkheda, Gujarat, India,
[2]Professor, Department of Computer Engineering, Government Engineering College, Rajkot, Gujarat, India,

Email ID: hetaljoshiyara@gmail.com[1], chiragthakerster@gmail.com[2]

Received: 14 March 2020 Revised and Accepted: 8 July 2020

ABSTRACT: Nowadays, Big Data (BD) has taken over many kinds of business. Since even a small scale company generates lots of data, technology is required to meet the speed of data generation, which can store and also analyze the huge quantity of data within a lesser time. And for such larger-scale data processing (DP), MapReduce (MR) is a largely employed parallel computing framework. In this paper, speculative execution (SE) can be utilized to enhance BD processing performance. This can well be stated as a common technique to take care of the straggler machine issue by means of plainly backing up those slower-running tasks on other machines. A meticulous analysis of scenarios is given along with that a new approach is developed, explicitly Supreme Rate Performance (SRP), which enhances the SE's effectiveness ominously. To exactly and also punctually recognize stragglers, a model is rendered that will identify which task to be backed up that correctly uses cluster resources. To calculate Remaining Time (RT) of task and predict processing speed we used modified exponentially weighted moving average. To select appropriate worker nodes (WN) intended for backup tasks, the Data Locality (DL) along with fairness are regarded. SRP is estimated in a physical homogeneous cluster and a Cloud-centered heterogeneous cluster. Experiment outcomes exhibit that in the presence of dead node or slow task SRP can run jobs faster up to 19 percent in heterogeneous Cloud cluster than homogeneous physical cluster using Hadoop-0.21.

KEYWORDS: MapReduce; straggler node; speculative execution; cluster throughput; Supreme Rate Performance.

I. INTRODUCTION

MR architecture [1, 2, and 3] was proffered by Google in 2004, and now, it has turned out to be a famous parallel computing framework for larger-scale DP [4, 5]. MR, as its name indicates, it splits into '2' functions – 1) Map Function 2) Reduce Function. The Map task's input is a compilation of data, and it transmutes it into another compilation of data, wherein individual constituents are broken into tuples (key-value pairs) as an output [6, 7]. The total execution process of the MR task is controlled chiefly by two entities. They are a) **Jobtracker**: which functions as a master node that is accountable for the total execution of the submitted job b) **Multiple Task Trackers**: which acts as a slave node, each of them doing the job. For each submitted job for execution in the system, there is one Jobtracker that exists on the Name node, and there are manifold task trackers present on Datanode [8]. When a system consumes an abnormally longer time to finish a task, it is acknowledged as straggler machine, it will hold-up the JET (the time of job initialized to that job retired) as well as demean the cluster throughput (the total jobs finished per second on the cluster) considerably [9]. This issue is managed by means of SE—the slower task is placed on a substitute machine with the intention that the backup can end quickly.

While distributing the load to every WN, there is a possibility of unequal load distribution, which brings data skewness. The data skew alludes to the disparity on the quantity of data allotted to every task or the disparity on the quantity of work needed for processing the data. In MR, skew [10] occurs when one node has more data allotted to be processed compared to others on the Map or Reduce phase [11]. Skew that might come as an irregular data distribution or DP cost generates stragglers, tasks that run considerably slower compared to others. These sluggish tasks could take above '5' times more to finish compared to the fastest task [3], in-turn slowing down the job completion. Here, a new standpoint is proposed for handling the skew issue on Hadoop [12, 13] applications with the subsequent objectives

✓Identify the slow task for SE(Backup/Rerun)

▫Finding out the RT of task

- ▫Estimate the backup time of task
- ▫Identify the proper node for backup(SE)
- ▫Handling Cluster Computing resources

✓Optimize BD [14, 15, 16, 17, and 18] processing in terms of

- ▫Improve job execution time (JET)
- ▫Cluster throughput

BACKGROUND

In this section, an MR description, the summary of the causes of stragglers, and after that, the internal mechanisms of some extensively utilized SE strategies were discussed.

MapReduce Mechanisms

In an MR cluster, subsequent to a job submitted, a master splits the inputted files into manifold Map Tasks (MT). Next, it schedules the MT, followed by reducing the tasks to WN. MT extract keys-value pairs as an input, transport it to some user-stated map function and joint function, and lastly produce the intermediary map outputs [19]. Then, the Reduce Tasks (RT) copy their inputted pieces in every map task, merged these to a single ordered (key, value list) pair stream via a merger sort, transport the stream to some user-stated reduce function, lastly, produce the outcome for the job.

Causes of straggler

The reasons for stragglers are categorized as i) internal ii) external as exhibited in Table 1. Internal can well be resolved by the MR Service provider, while external reasons can't. The "internal reason" is shunned by means of restraining every WN to run at-most one task concurrently or by merely allowing tasks with disparate resource usage intensity to share the same WN. Nevertheless, resource competition can't be attributed to co-hosted applications as there is no control over other users' VM. Amongst these causes, the resource capacity heterogeneity of WN is typically stable and also predictable, while others are not. For the majority of these factors, SE is an effectual way to resolve the straggler issue.

Table 1 Causes of straggler

Internal Factors (Categorized by behaviour addressed by MapReduce Service Provider)	**External Factors (User behaviour and System Environment)**
Block size and Slot number [22]	unexpected Surge on User Demand [22]
Limited processing power [22]	Poor user code , Faulty hardware[20]
Heterogeneous node capacities [20]	CPU overload, heterogeneous node capacities[22]
Resource competition due to other MR tasks running on the same WN[20]	Physical nodes can host an unknown number of virtual machines(Cloud Environment)[21]
Resource starvation in Cloud Environment due to virtual machine[21]	Skewed input data size, Queuing, Power Limits [20]
	Unbalanced workload, Shared resource contention [21] Network Bandwidth, I/O activities [21]

Therefore major causes of straggling are BLOCK SIZE partition, uneven File Size (FS), Load imbalance, Poor user code, and Faulty Hardware may lead to Straggler.

II. LITERATURE SURVEY

Several SE strategies were presented in the literature, encompassing the original Hadoop [23], MR [24], Wrangler [25], together with LATE [26]. These only begin the SE when the job is almost done and also the cluster has available task slots. Some recent studies concentrate on enhancing the SE [27], [28], and [29].

Jeffrey Dean and Sanjay Ghemawat [24] employed the MR programming model at Google for numerous disparate reasons. This design was utilized for various reasons. Initially, the model was simple to utilize, even for programmers devoid of experience with a parallel in addition to distributed systems, as it concealed the details of parallelization, locality optimization, fault tolerance, together with load balancing. Next, a large diversity of issues was effortlessly expressible as MR computations. Followed by, an MR implementation was developed that scaled to huge clusters of machines enclosing thousands of machines. The implementation made effective utilization of these machine resources, and thus, was appropriate for utilization on many of the big computational issues that were encountered at Google.

N. J. Yadwadkar et al. [25] envisaged which task benefited as of SE and begun speculative tasks correspondingly. Nevertheless, the prediction depended upon a statistical learning design that was derived as of historical workloads. The performance intrusion on the cloud affected the prediction's accuracy.

Matei Zaharia et al. [26] examined the issue of SE in MR. The author found the faults with the specific threshold-centered scheduling algorithm in Hadoop, in addition to, progress-rate-centered algorithms generally. A plain, tough scheduling algorithm, along with Longest Approximates Time to End (LATE) was modeled that employed estimated ending times for speculatively performing the tasks that harmed the response times mostly. The technique was enhanced than Hadoop's default SE algorithm.

Ananthanarayanan et al. [27] proffered a system for monitoring tasks and choosing outliers termed Mantri utilizing resource- and cause- aware approaches. Mantri's approaches encompass shielding outputs of imperative tasks, re-starting outliers, and network-aware task placement. Utilizing practical reports, Mantri recognized and served the outliers earlier on its lifetime. Acting grounded on the causes, the opportunity resource, and cost of actions allow Mantri to ameliorate over prior work that just duplicated the laggards. Employment in trace-driven simulations along with Bing's production clusters confirmed that Mantri improved JET by 32%.

Chen et al. [28] rendered a Maximal Performance Cost approach that deployed an Exponentially Weighted Moving Averages (EWMA) for predicting the RT of a task. MCP picked the node intended for SE utilizing a complicated cost-effective framework that regarded the DL and the workload on every node. It highly ameliorated job performance. The approach could not recognize the performance tailback of the straggler tasks and can't assure that the tasks' speculative copies would do better**.**

Shadi Ibrahim et al. [29] suggested elevating the MR computation's performance utilizing a scheduling algorithm aimed at MT, termed Maestro. The suggested Maestro scheduled the MT in the succeeding manners: Initially, the empty slots of every Data Node (DN) is filled grounded upon the input data's replication strategy and total hosted MT. Secondarily, the runtime schedule regarded the likelihood of scheduling a MT on a provided machine reliant on the duplications of the task's inputted data. In an experiment with a 100node cluster, the suggested Maestro algorithm attained 95 percent local map execution, brought 34% execution time (ET) improvement, and produced 80%, reduction in the SE MT as well.

Quan Chen et al. [30] suggested the Self-Adaptive MR (SAMR), which automatically adapts a continually varying environment, to evaluate a tasks' progress dynamically. As a job is submitted, SAMR divided the jobs into countless fine-grained maps, lessen tasks and then assign them to innumerable nodes. Moreover, it interpreted historical (past) information of the nodes and even updated it after each execution. Subsequently, SAMR adjusted the time weights of every map-stage and lessen tasks accordingly. Hence, it accurately attained the progress in a task and determined which tasks required backup tasks. Outcomes evinced that SAMR notably diminishes the ET up to 25 percent contrasted to Hadoop's scheduler as well as up to 14 percent when contrasted with the LATE. Nevertheless, the algorithm failed to concentrate on DL while initiating backup tasks, since it might extraordinarily speed-up the data loading together with storing.

Drawbacks of Existing Methodologies

In Hadoop original, the existing algorithms have the backup candidate's selection as a drawback. LATE algorithm regarded the average progress rates of map, shuffle, and reduce phase to pick slow tasks and estimate their RT. They were grounded on the following assumptions:
i) Map, Shuffle, and Reduce phases consume the same sum of time for roughly processing the same volume of inputted data.
ii) Progress rate should be stable or speed-up amid a task's life span. The drawbacks prevailing in the existing algorithms are briefly explicated as follows

In selecting backup candidates

Input Data Skew

The tasks don't process the same volume of data at all times and they might experience numerous sorts of data skew on MR and it was examined by Kwon **et al.** [31]. As the input data could not divide the big records, the MT, which process innumerable data, would process such records. As the key distribution in the inputted data set is skewed, the uneven partition of the intermediary data created by the MT would also bring the partition skew amongst the RT. Hence, the initial assumption could break effortlessly and this was also examined in [32].

Phase Percentage Not Matching Corresponding Duration Ratio
The subsequent assumption is performed via LATE for simplifying the computation of a task's RT, however practically it might fail. For demonstrating this, time duration is examined for all phases in Hadoop aimed at typical jobs, like Grep, Sort, and Word Count, on the local cluster.

Reduce Tasks Starting Asynchronously before All Map Tasks Complete
The aforesaid secondary assumption might as well fail once the RT is asynchronously launched before on account of inadequate free slots on a cluster. In this scenario, they would copy the produced map outputs quickly when some RT is launched subsequent to a fraction of completion of the MT. But then, they need to wait for the MT to produce new-outpu. It slows-down the Process Speed (PS) of the copy stage and even breaks the aforesaid secondary assumption.

Using std as SlowTaskThreshold
Hadoop-LATE inappropriately utilizes std as the slowTasks Threshold; at times, it miscalculate that there are lots of stragglers or can’t recognize any at all.

Pitfalls in Selecting Backup Worker Nodes
Identifying Slow Worker Nodes Improperly
For recognizing the straggler nodes, LATE together with Hadoop-LATE utilize a threshold say, slowNodeThreshold. Both regard a WN as a slow node when its performance score is lower compared to the average performance score of the entire nodes by means of a threshold. In addition, they would, for no reason, give any tentative task to this slower node. Nevertheless, certain WN might do more tedious tasks and unfairly attain lower performance scores. Consequently, such WN are concerned to be slow.

Choosing Backup Worker Nodes Improperly
Both LATE and Hadoop-LATE do not utilize DL to verify if backup tasks could finish early while picking backup nodes. They presume that network consumption is adequately lower at the time of the map phase since maximal MT is data-local. Consequently, they presume that the non-local MT could run faster similar to data-local ones. Moreover, it is perceived that the ET of a data-local MT could be about 3 times quicker on considering the ET of a non-local MT, which motivates to regard DL.

In finding straggler node
Takes a long Time to Identify Stragglers
As users share all MR clusters, the WNs’ performance may be degraded just because other users have launched certain applications. This slows down some tasks spectacularly. Unluckily, it would consume a longer time for them to be recognized as stragglers by means of utilizing the average PS which will fall to 2 percent per second just after 53b sec. Consequently, the RT evaluated grounded on the average PS would be much little on considering the original value. Hence, utilization of this average PS could not recognize the straggler tasks on time and it could even bring misjudgements. The limitations prevailing in the existing algorithms could be comprehended using fig. 1.

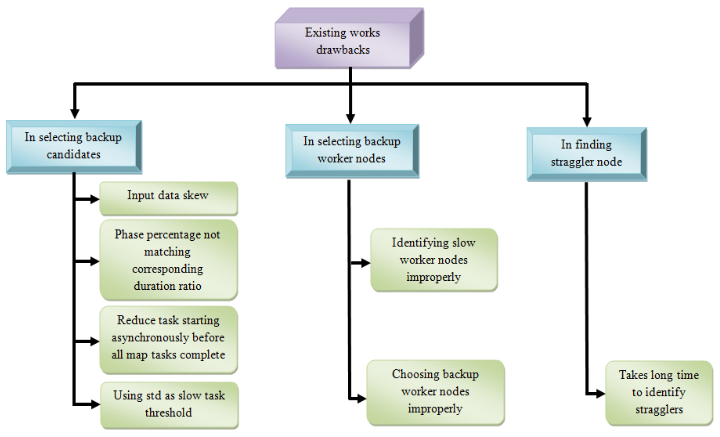

Existing works drawbacks
In selecting backup candidates
In selecting backup worker nodes
In finding straggler node
Input data skew
Phase percentage not matching corresponding duration ratio
Reduce task starting asynchronously before all map tasks complete
Using std as slow task threshold
Identifying slow worker nodes improperly
Choosing backup worker nodes improperly
Takes long time to identify stragglers

Fig. 1. Limitations of the existing works

III. PROPOSED METHODOLOGY

Here, a new SE strategy called SRP is proposed aimed at the utmost cost performance. Every task running on disparate Datanode is monitored for every phase (explicitly Map, Shuffle and Reduce). The cost is regarded to be the computing resources encompassed via tasks, whilst the performance to be the shortening of JET and the augmentation of the cluster throughput. SRP aims at choosing straggler tasks precisely as well as promptly and also backing them up on appropriate WN. To make sure fairness, task slots are allotted in the order the jobs are given. Similar to other SE, SRP renders new tasks a high priority compared to backup tasks. To put it differently, until the SRP has assigned all the new MT/RT, it will not begin backing up straggler MT/RT. Centred on a prompt forecast of the tasks' PS along with a precise assessment of their RT, the SRP selects backup candidates. Next, these backup contenders will be backed up selectively on appropriate WN to attain maximal cost performance as per the cluster load. In this part, the proposed techniques' pseudo-code intended for estimating remaining task time, backup time of slow tasks, as well as identification of the proper node for back up are given meticulously, and the SRP's result and implementation are exhibited in the result and discussion section.

Backup Candidates Selection

Modified Exponentially weighted moving average (EMWA) for predicting process speed

In SRP, rather than merely utilizing the past average rate for predicting the tasks' PS, modified EWMA is being used. The EWMA alludes to an average of data that is utilized to track the portfolio's movement by means of checking the outcomes as well as output via considering the disparate factors, additionally, providing them the weights, and then, tracking outcomes to evaluate the performance and also to make enhancements**.** There are numerous prediction algorithms, like EWMA, CUSUM [33**],** and modified EWMA **[**34**]** to envisage the tasks PS. Here, a modified adaptation of the EWMA is chosen. The EWMA suffers as an inertia issue on account of 'error' in the EWMA statistic. The Modified EWMA regards precedent observations same as the EWMA, in addition, regards precedent changes, along with the newest modification in the procedure. A Modified EWMA is exhibited as:

$$X(t) = (1-\alpha)X(t-1) + \alpha Y(t) + (Y(t) - Y(t-1)) \quad (1)$$

Wherein, α implies a suitable constant, in order that $0 < \alpha \leq 1$, $X(t)$ as well as $y(t)$ signify the estimated as well as the observed PS at time t **[**16**]**. Here, α implies a tradeoff betwixt stability and responsiveness. In this, α is fixed to 0.2 as per **[**16**].** α is the degree of mixing parameter value between 0 and 1. If α =1 that implies only the freshest data has been utilized to gauge EWMA. If α is approaching 0, that implies more weightage is rendered to old data, and if α is close to 1 that implies new data has been rendered more weightage **[**35**]**. The experiment is performed utilizing EMWA as well as Modified EWMA and RMS. While computing the estimated time, the difference between speed recorded for the current and the preceding time cycle is added. Observing the graph shows that the modified EWMA envisages the RT of the process which is closer to the recorded observed process

$$X(t) = (1-\alpha)X_{n-1} + \alpha Y_n \quad (2)$$

Estimation of Remaining Task Time

In SRP, the highest priority for back-up is proffered to a task having the longest RT. As said before, the sum of the RT left in every phase is concerned as a task's RT. When a task is running on the current phase ($curr\_phase$), the RT left on $curr\_phase$ is estimated by the processing bandwidth in $curr\_phase$ and the factors of the remaining data. Nevertheless, the calculation of the RT of the following phases $follow\_phase$ is hard as the task hasn't entered those phases yet. So, the phase average PS is utilized for estimating the RT of a phase Avg_{phase}. The average PS of tasks that entered the phase is regarded as the phase's average PS. Aimed at those phases where no task has entered, their RT is not evaluated (for the fair consideration of all tasks). As tasks might process a disparate volume of data, Avg_{phase} is attuned by factor μ that signifies the ratio of the inputted size of this task to the average of all tasks' inputted sizes. The steps for estimating the RT of task are enumerated and explained below.

Step 1: Declare Variables

Step 2: Evaluate the RT of task as

$$RT_{task} = RT_{curr\_phase} + RT_{follow\_phase}$$

$$= RDP_{curr\_phase} \Big/ \mathrm{BWR}_{\mathrm{curr\_phase}} + \sum Avg_{phase} * \mu_{phase} \quad (3)$$

in follow_phase

Where RT denotes the Remaining Time, RDP signifies Remain_DataProcessed, and BWR specifies the BandwidthReq

Step 3: Evaluate the constant μ utilizing the equation (4)

$$\mu = DV_{input} \Big/ DV_{Avg} \quad (4)$$

Step 4: Determine the RT utilizing the equation (5) as follows

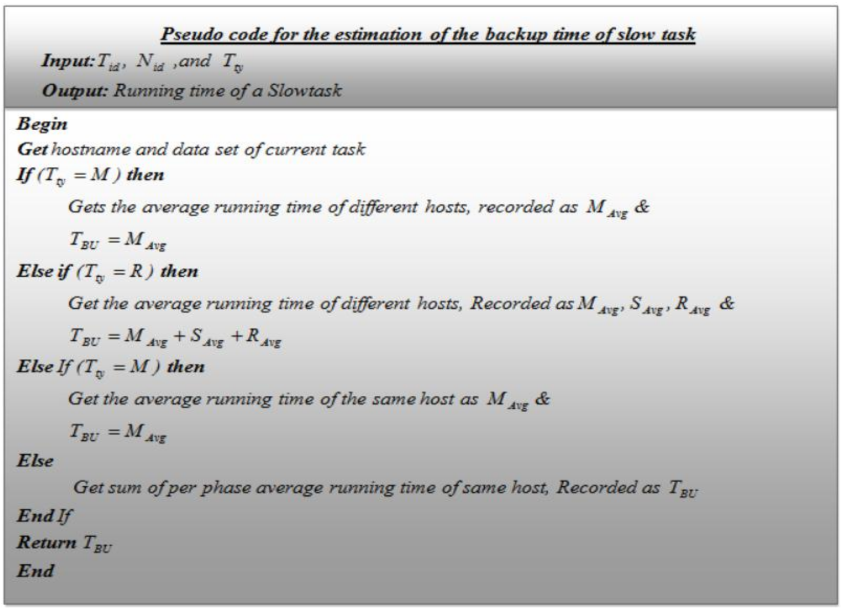
$$RT_{shuffle} = \left[PF_{map} - PF_{shuffle} \right] \div \mathrm{PS}_{\mathrm{shuffle}} \quad (5)$$

Where, PF indicates the Percent_Finish, and PS signifies the Process speed

Step 5: If the task has the longest RT, then that have highest priority for SE (Back Up)

Estimation of the Backup time of slow task

The system checks the succeeding conditions to estimate the backup time of a slow task. If the dataset is empty and phase type is Map, then the backup time is equivalent to the Map Avg; else if the phase type is Reduce, then the task backup is evaluated by summing Map Avg, shuffle Avg and reduce Avg; else if the phase type equals Map, then the Task backup time is equal to Map Avg; else, sum each phase's average running time of the same host. Here, the steps performed in the proposed methodology for estimating the Backup time of slow task are proffered as a pseudo-code, which is shown using fig. 2.



```
Pseudo code for the estimation of the backup time of slow task
Input: $T_{id}$, $N_{id}$, and $T_{ty}$
Output: Running time of a Slowtask
Begin
Get hostname and data set of current task
If ($T_{ty} = M$) then
        Gets the average running time of different hosts, recorded as $M_{Avg}$ &
        $T_{BU} = M_{Avg}$
Else if ($T_{ty} = R$) then
        Get the average running time of different hosts, Recorded as $M_{Avg}$, $S_{Avg}$, $R_{Avg}$ &
        $T_{BU} = M_{Avg} + S_{Avg} + R_{Avg}$
Else If ($T_{ty} = M$) then
        Get the average running time of the same host as $M_{Avg}$ &
        $T_{BU} = M_{Avg}$
Else
        Get sum of per phase average running time of same host, Recorded as $T_{BU}$
End If
Return $T_{BU}$
End
```

Fig. 2. Pseudocode for the estimation of backup time of the slow task

In the pseudocode,

$T_{id}, N_{id}, and\ T_{ty}$ - TaskId, NodeId, and TaskType

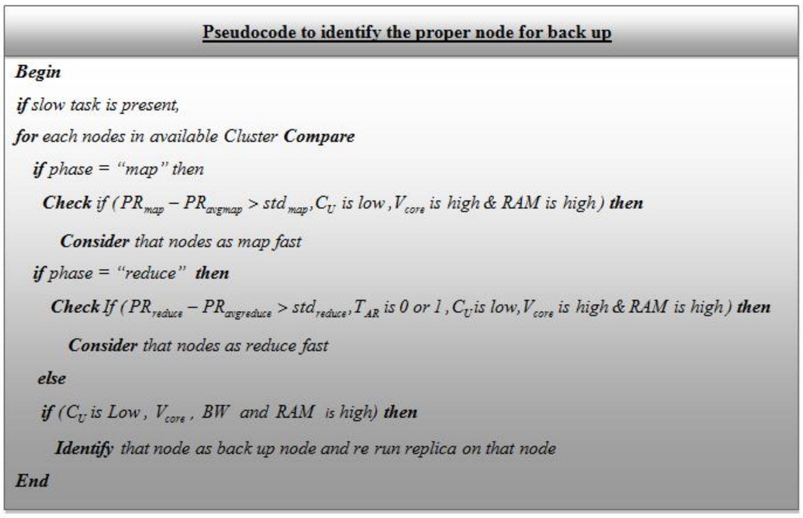
$M, R, and\ S$ - Map, Reduce and Shuffle

T_{BU} - Task Backup Time

$M_{Avg}, R_{Avg}, and\ S_{Avg}$ - Map Average, Reduce average, and Shuffle Average

Pseudocode to identify the proper node for back up

It is essential to assign backup tasks to fast WNs for getting better performance. This needs a relevant metric to effectively gauge the WNs' performance, which highly varies from time to time. For instance, Microsoft witnessed that slow WNs vary over weeks on account of altering data popularity. The LATE and Hadoop-LATE could not accurately assess the WNs' performance. For tackling this problem, the moving averages procedure bandwidth of data-local MT is utilized, which is completed by a WN to signify the node's performance. The Pseudocode for selecting a proper backup node is proffered using fig. 3.



Pseudocode to identify the proper node for back up

```
Begin
if slow task is present,
for each nodes in available Cluster Compare
  if phase = "map" then
    Check if ($PR_{map} - PR_{avgmap} > std_{map}, C_U$ is low, $V_{core}$ is high & RAM is high) then
      Consider that nodes as map fast
  if phase = "reduce" then
    Check If ($PR_{reduce} - PR_{avgreduce} > std_{reduce}, T_{AR}$ is 0 or 1, $C_U$ is low, $V_{core}$ is high & RAM is high) then
      Consider that nodes as reduce fast
  else
  if ($C_U$ is Low, $V_{core}$, BW and RAM is high) then
    Identify that node as back up node and re run replica on that node
End
```

Fig. 3. Pseudocode for identifying the proper node for back

In the Pseudocode,

PR -ProgressRate

C_U - CPU Utilization

T_{AR} - Total Time Spent by all reduce waiting after reserving slots (ms)

IV. RESULT AND DISCUSSION

The proposed ameliorated speculative approach is executed via establishing the physical homogeneous Hadoop Cluster (HC) as well as cloud-centered HC with the subsequent configurations (Table 2). For the proposed work's implementation, the HC was performed utilizing the AWS cloud system. The configurations of the Name Node as well as DN are exhibited in table 2, and the configurations of datasets along with its sorts for executing a range of programs are exhibited in table 3. The outcome achieved by means of the '6' nodes cluster and also '7' nodes cluster are exhibited in Figures 4 and 5.

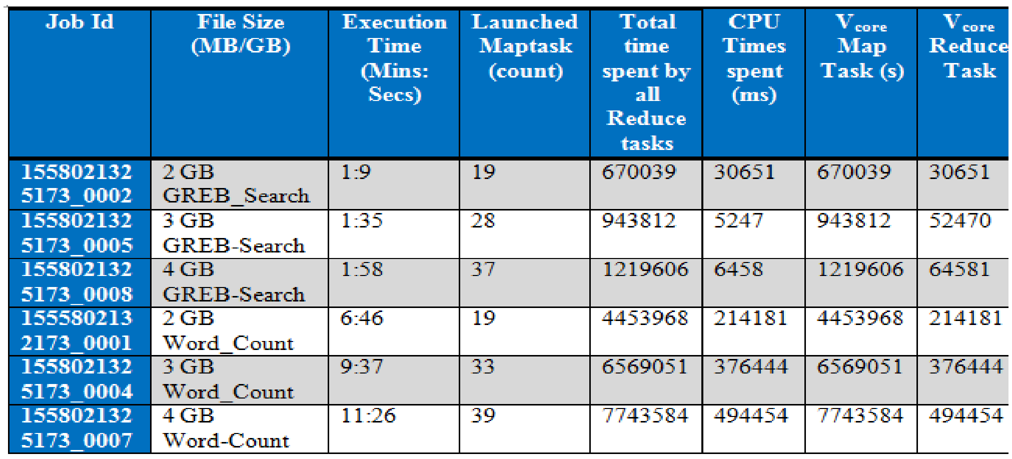
Table 2 Configurations of Name Node & DN

Particulars	Name Node (Job Tracker)	Data Node (Task Tracker)
Processor	Intel Xeon processors	Intel Xeon processors
RAM	1 GB	1 GB
Hard Disk	105.8 GB	105.8 GB
OS	Ubuntu 17.10	Ubuntu 17.10
JDK Version	Jdk1.8.0-171	Jdk1.8.0-171
HADOOP	2.6.5	2.6.5
Replication Factor	3	3
Eclipse	OXYGEN.3A For 64- bit	OXYGEN.3A For 64- bit

ISSN- 2394-5125 VOL 7, ISSUE 15, 2020

Table 3 Datasets & its types for executing various programs

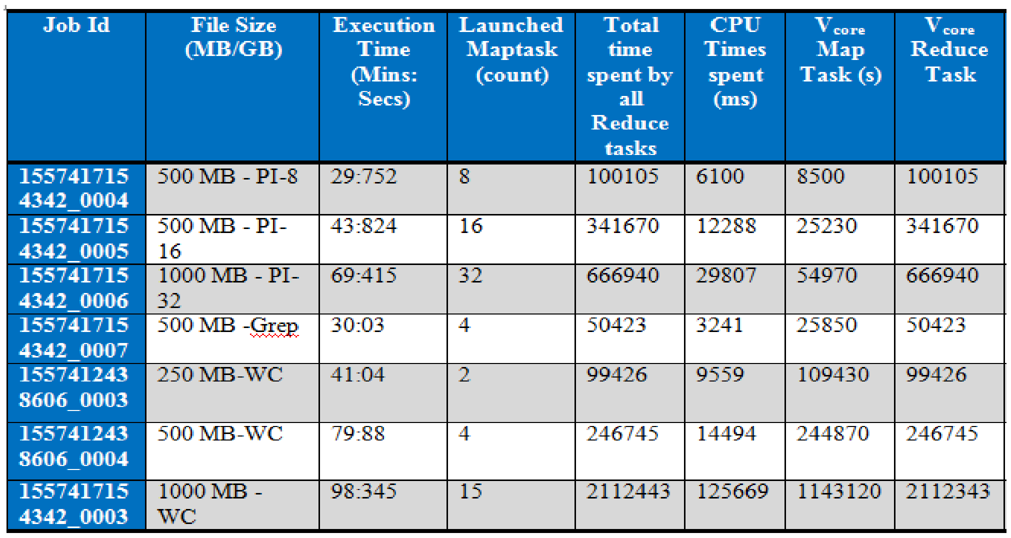
Program	Dataset	Type
Word Count	Wikipedia	CPU Intensive
GrepSort	Wikipedia	CPU Intensive
GrepSearch	Wikipedia	CPU Intensive
Terasort	Generate using Teragen	I/O Intensive
TestDFSIO	Generate using Write Method	I/O Intensive
PI	Number of Maptasks and Sample	CPU Intensive



Job Id	File Size (MB/GB)	Execution Time (Mins: Secs)	Launched Maptask (count)	Total time spent by all Reduce tasks	CPU Times spent (ms)	V_{core} Map Task (s)	V_{core} Reduce Task
155802132 5173_0002	2 GB GREB_Search	1:9	19	670039	30651	670039	30651
155802132 5173_0005	3 GB GREB-Search	1:35	28	943812	5247	943812	52470
155802132 5173_0008	4 GB GREB-Search	1:58	37	1219606	6458	1219606	64581
155580213 2173_0001	2 GB Word_Count	6:46	19	4453968	214181	4453968	214181
155802132 5173_0004	3 GB Word_Count	9:37	33	6569051	376444	6569051	376444
155802132 5173_0007	4 GB Word-Count	11:26	39	7743584	494454	7743584	494454

Fig. 4. Results of the 6 node cluster

Fig. 4 exhibits the outcome attained by means of the six node cluster. In this figure, the disparate attained outcomes of disparate job application Ids are shown. The job Ids with disparate FS are regarded for the experimentation. The outcomes of ET, launched Maptask, total time spent by the lessened tasks, CPU times spent, V_{core} Maptask, V_{core} Reduce Task are provided. Here, the ET of the job Id 1558021325173_005 (2 GB GREB Search) is (1:9), which is lesser when contrasted with the other job Ids. Aimed at this id, the remaining measures, for instance, launched Maptask (19), total time spent by all RT (670039), CPU time spent (30651), V_{core} Maptask (670039), and V_{core} Reduce Task (30651) renders the lowest values when weighted against other job Ids. The metric values are changed as per the FS used by the job Id s.



Job Id	File Size (MB/GB)	Execution Time (Mins: Secs)	Launched Maptask (count)	Total time spent by all Reduce tasks	CPU Times spent (ms)	V_{core} Map Task (s)	V_{core} Reduce Task
155741715 4342_0004	500 MB - PI-8	29:752	8	100105	6100	8500	100105
155741715 4342_0005	500 MB - PI-16	43:824	16	341670	12288	25230	341670
155741715 4342_0006	1000 MB - PI-32	69:415	32	666940	29807	54970	666940
155741715 4342_0007	500 MB -Grep	30:03	4	50423	3241	25850	50423
155741243 8606_0003	250 MB-WC	41:04	2	99426	9559	109430	99426
155741243 8606_0004	500 MB-WC	79:88	4	246745	14494	244870	246745
155741715 4342_0003	1000 MB - WC	98:345	15	2112443	125669	1143120	2112343

Fig. 5. Results of the 7 node cluster

ISSN- 2394-5125 VOL 7, ISSUE 15, 2020

Fig. 5 exhibits the outcomes of the seven node cluster with seven disparate job Ids. These job ids utilize the '7' disparate FS for their execution. Here, the ET of the 1557417154342_0004 is 29:752 that is lower when weighed against the other job Ids. In addition, for all the rest of the metrics, this job id comes with the least values, which utilizes the FS of 500 MB-PI-8. The job id 1557412438606_0004 renders the highest ET of 98:345 when utilizing the 1000MB-WC file size. The Launched MT of the 1557417154342_0007 job Id is highest (32) amid all others, which utilizes the FS of 1000 MB PI-32. Here, the seven disparate job Ids utilizes seven disparate FS, which gives disparate values for all the metrics.
The technique's ET utilizing disparate FS is exhibited in figure 6. The FS utilized are 500 MB PI-8, 500 MB PI-16, 1000 MB PI-32, 500 MB-Grep, 250 MB-WC, 500 MB-WC, along with 1000 MB WC, which renders the ET of 29:752, 43:824, 69:415, 30:03, 41:04, 79:88, 98:345. In addition, the Maptask time of these same job Ids with disparate FS is exhibited in figure 7. Here, the Maptask time of the 1000 MB-WC is 2250000, which is highest when weighted against all other FS. And the 500 MB PI-8 renders the smallest Maptask time when weighted against other FS in the technique.

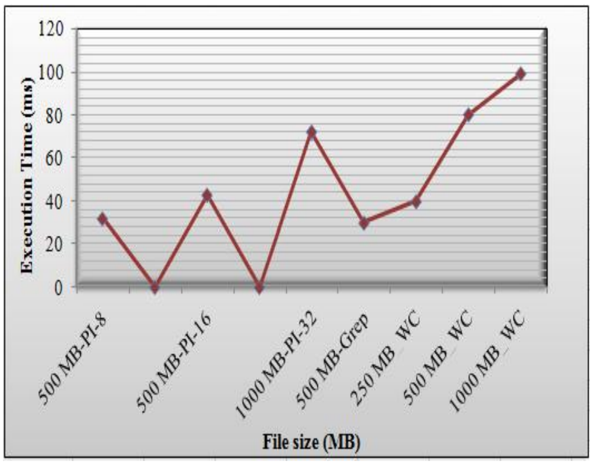


Fig. 6. ET of the technique for different FSs

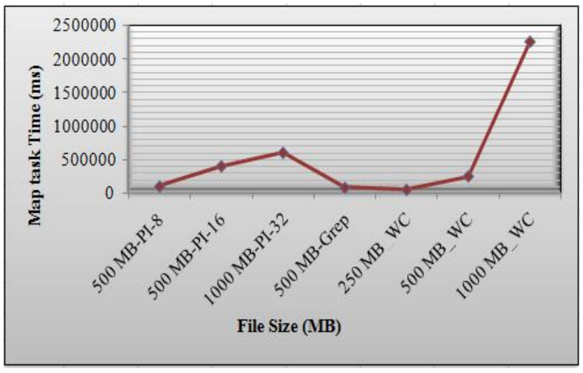


Fig. 7: MT Time of the techniques with different FS

Fig. 8 illustrates the reduce task time of disparate FS. For the 500MB PI-8 FS, the method gives the reduced task time of 4000ms, which is low when contrasted to the reduce task time of other FSs. The 500MB PI-16 FS offers the higher reduce task time on considering the 500MB PI-8 FS, which is 16000ms. The reduce task time varies for all different FSs. Here, the reduce task time of 1000 MB_WC file is larger (130000) when contrasted to all other FSs. And the 3 FSs, namely 500 MB-GREP, 500 MB-WC, and 500 MB-PI-8 offer the lowest reduce task time of 1000ms, 10000ms, and 4000ms amongst all others. The ET consumed by the proposed and existing approaches is proffered in fig. 9.

ISSN- 2394-5125 VOL 7, ISSUE 15, 2020

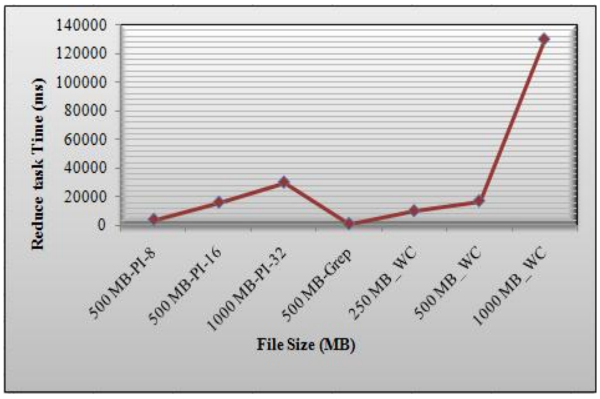


Fig. 8. Reduced task time of the techniques with different file sizes

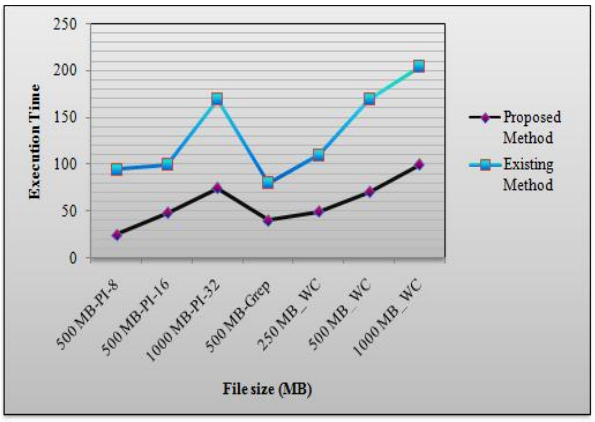


Fig. 9: Execution time of the proposed and existing techniques

Fig. 9 regarded seven different FSs for this comparison. Here, the ET is gauged in milliseconds. For the 500 MB-PI-8 FS, the existing method takes 95ms ET, whereas the proposed technique takes 25ms ET, which is extremely low when contrasted to the existing algorithm. Likewise, for the remaining FSs, namely 500 MB PI-16, 1000 MB PI-32, 500 MB- Grep, 250 MB_WC, 500 MB_WC, 1000 MB_WC, the proposed method takes 48, 75, 40, 49, 70, 100(in milliseconds) which is lower on concerning the existing algorithm, which shows the proposed method's efficiency. The remaining process time of the existing and proposed approaches by varying the time samples from 10 to 20 is proffered in fig. 10.

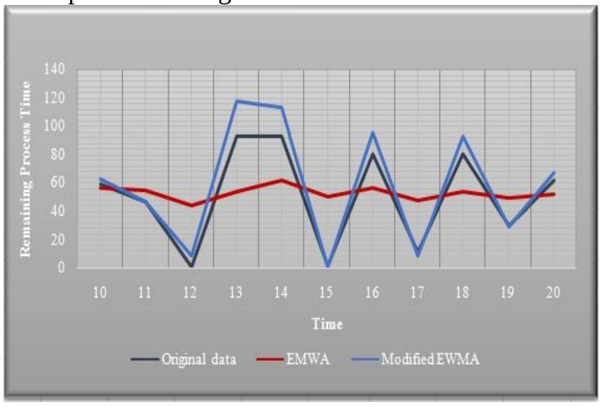


Fig. 10 Remaining process time of the techniques by varying the time from (10 to 20)

Fig. 10 shows the remaining process time of the system using original data, existing EMWA, and proposed modified EMWA. For the10ms time sample, the
remaining process time of the system for the original data is 59.0628, whereas, the existing EMWA gives 56.3073, and the proposed modified EMWA offers 62.3799 which is higher when contrasted to the original data and the existing EMWA. The proposed modified EMWA has low processing time for certain time samples. But for at most time samples, the proposed modified EMWA attains higher process time when contrasted to original data and existing methodologies. The remaining

ISSN- 2394-5125 VOL 7, ISSUE 15, 2020

process time of the techniques by varying the time samples from 0 to 148 is proffered in fig. 11. The time gap between the samples is 4 and the processing time is plotted for those time samples. In the graph, the proposed EMWA reveals the highest level of remaining process time for the utmost time samples when contrasted to the processing time of the original data and existing EMWA.

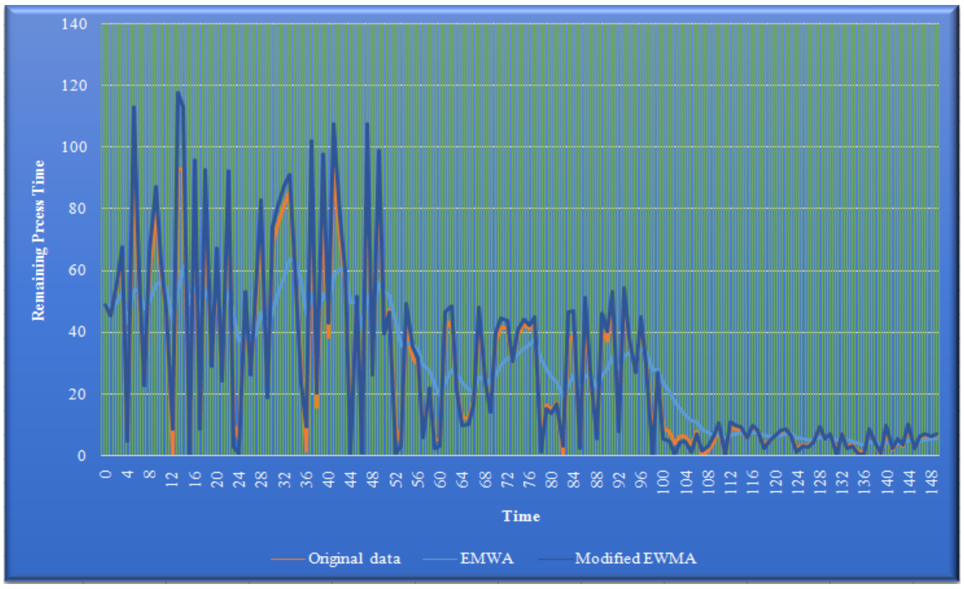


Fig. 11. Remaining Process Time of the techniques by varying the time from (0 to 148)

The average time consumed by the proposed system and existing system is 56.04086 and 75.04086 accordingly, which reflects that the proposed system is having excellent results in respect of ET (about 19%) while considering the existing system. The proposed system utilizes the AWS cloud system for implementing the HC, as provided in table 3. It is obvious that the cloud performs well on considering the physical HC set-up in the laboratory. It is beneficial in many terms to utilize the cloud in place of the physical system. Table 3 proffers a comparison of the same. It is inferred that irrespective of job type (CPU intensive or i/o intensive), the shuffle phase consumes a longer time, where the theoretical assumption is that the map, shuffle, and reduce consume the same amount of time
.

Table 4 Comparison Of Physical Cluster& Cloud Cluster Using Wordcount Program

		Comparison			
S. No	File Size	Physical Hadoop Cluster	Cloud Hadoop Cluster	Pysical Hadoop Cluster (4 live nodes and 1 dead node)	Cloud Hadoop Cluster (4 live nodes and 1 dead node)
		Execution time (Mins: Seconds)	Execution time (Mins: Seconds)	Execution time (Mins: Seconds)	Execution time (Mins: Seconds)
1	250 MB	01:38	0.684	1.4637	1.23
2	500 MB	02:45	1.33	2.0575	1.729
3	1 GB	05:43	1.63	3.2975	2.771
4	2 GB	10.47	6.46	9.9936	8.398
5	4 GB	14.82	11.26	17.4097	14.63

V. CONCLUSION

SE is the optimization approach which can be utilized for improving the JET in BD Processing Environment. If it is managed effectively, it would elevate the cluster's throughput and JET. It is perceived that the proposed system is 19% more efficient on considering the existing one as per table 4. The proposed system aims to transfer the system load to its efficient node which is elected grounded on its previous performance. During SE, the existing system transfers the load to its closest adjacent node, which lessens its efficiency. Here, it is identified that the average time consumed by the proposed system and the existing system is 56.04086 and 75.04086 respectively. Efficiency could be further ameliorated in the future by utilizing machine learning strategies for selecting nodes in SE. If the system is slow, one could predict where to transfer the system load to get better output and gain better efficiency utilizing machine learning algorithms.

VI. REFERENCES

[1] Mohammed Al-kahtani, S, and Lutful Karim, “An efficient distributed algorithm for big data processing”, Arabian Journal for Science and Engineering, vol. 42, no. 8, pp. 3149-3157, 2017.

[2] Qiufen Xia, Weifa Liang, and Zichuan Xu, “Data locality-aware big data query evaluation in distributed clouds”, pp. 791-809, 2017.

[3] F
eng Yu, Eric S. Jones, and Wen-Chi Hou, “Write optimization using asynchronous update on out-of-core column-store databases in map-reduce”, In IEEE International Congress on Big Data, IEEE, pp. 720-723, 2015.

[4] D
ean J and Ghemawat S, “Mapreduce: Simplified Data Processing on Large Clusters,” Comm. ACM, vol. 51, pp. 107-113, 2008.

[5] Liying Li, Zhuo Tang, Renfa Li, and Liu Yang, “New improvement of the Hadoop relevant data locality scheduling algorithm based on LATE”, In International Conference on Mechatronic Science, Electric Engineering and Computer (MEC), IEEE, pp. 1419-1422, 2011.

[6] Xu-qing Chai, Yong-liang Dong, and Jun-fei Li, “Profit-oriented task scheduling algorithm in Hadoop cluster”, EURASIP Journal on Embedded Systems, no. 1, pp. 1-8, 2016.

[7] Liya Thomas, and R. Syama, “Survey on MapReduce scheduling algorithms”, International Journal of Computer Applications, vol. 95, no. 23, 2014.

[8] https://cwiki.apache.org/confluence/display/HADOOP2/TaskTracker

[9] A
shwin Bhandare, Jitin George, Supreet Deshpande, and Yash Karle, “Review and analysis of straggler handling techniques”, Int. J. Comput. Sci. Inf. Technol, vol. 7, no. 5, pp. 2270-2276, 2016.

[10] Dhanalakshmi S, Arputhamary B, Arockiam L, “A Survey on Data Skew Mitigating Techniques in Hadoop MapReduce Framework”, International Journal of Computer Science and Mobile Computing, vol. 5, no. 7, pp. 290-296, 2016.

[11] Kwon Y, Balazinska M, Howe B, and Rolia J, “SkewTune: Mitigating skew in mapreduce applications,” in Proc. ACM SIGMODInt. Conf. Manage. Data, 2012, pp. 25–36.

[12] Guilherme Cassales W, Andrea Schwertner Charao, Manuele Kirsch-Pinheiro, Carine Souveyet, and Luiz-Angelo Steffenel, “Improving the performance of Apache Hadoop on pervasive environments through context-aware scheduling”, Journal of Ambient Intelligence and Humanized Computing, vol. 7, no. 3, pp. 333-345, 2016.

[13] Brahmwar, M., M. Kumar, and G. Sikka, “Tolhit–a scheduling algorithm for hadoop cluster”, Procedia Computer Science, vol. 89, pp. 203-208, 2016.

[14] Uthayasankar Sivarajah, Muhammad Mustafa Kamal, Zahir Irani, and Vishanth Weerakkody, “Critical analysis of Big Data challenges and analytical methods”, Journal of Business Research, vol. 70, pp. 263-286, 2017.

[15] Amir Gandomi, and Murtaza Haider, “Beyond the hype: Big data concepts, methods, and analytics”, International journal of information management, vol. 35, no. 2, pp. 137-144, 2015.

[16] Uthayasankar Sivarajah, Muhammad Mustafa Kamal, Zahir Irani, and Vishanth Weerakkody, “Critical analysis of Big Data challenges and analytical methods”, Journal of Business Research, vol. 70, pp. 263-286, 2017.

[17] Ashley Braganza, Laurence Brooks, Daniel Nepelski, Maged Ali, and Russ Moro, “Resource management in big data initiatives: Processes and dynamic capabilities”, Journal of Business Research, vol. 70, pp. 328-337, 2017.

[18] Qinghua Lu, Shanshan Li, Weishan Zhang, and Lei Zhang, “A genetic algorithm-based job scheduling model for big data analytics”, EURASIP journal on wireless communications and networking, no. 1, pp. 152, 2016.

[19] Matei Zaharia, Dhruba Borthakur, Joydeep Sen Sarma, Khaled Elmeleegy, Scott Shenker, and Ion Stoica, "Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling", In Proceedings of the 5th European conference on Computer systems, pp. 265-278, 2010.

[20] Qi Chen, Cheng Liu, and Zhen Xiao "Improving MapReduce Performance Using Smart Speculative Execution Strategy", Member, IEEE Transactions on Computers, vol. 63, no. 4, 2014

[21] Yanfei Guo, Jia Rao, Changjun Jiang, and Xiaobo Zhou, "Moving Hadoop into the cloud with flexible slot management and speculative execution", IEEE Transactions on Parallel and Distributed systems, vol. 28, no. 3, pp. 798-812, 2016.

[22] 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 (TOIT), vol. 18, no. 2, pp. 1-22, 2018.

[23] Apache Hadoop Project. (2013). [Online]. Available: http://hadoop.apache.org

[24] Dean J and Ghemawat S, "MapReduce: Simplified data processing on large clusters," In Proc. 6th Conf. Symp. Operating Syst. Des.Implementation, 2004, no. 10.

[25] Yadwadkar N. J, Ananthanarayanan G, and Katz R, "Wrangler: Predictable and faster jobs using fewer resources," in Proc. ACMSymp. Cloud Comput, pp. 1–14, 2014.

[26] Zaharia M, Konwinski A, Joseph A. D, Katz R, and Stoica I, "Improving MapReduce performance in heterogeneous environments," in Proc. 8th USENIX Symp. Operating Syst. Des. Implementation, pp. 29–42, 2008.

[27] Ganesh Ananthanarayanan, Srikanth Kandula, Albert G. Greenberg, Ion Stoica, Yi Lu, Bikas Saha, and Edward Harris, "Reining in the Outliers in Map-Reduce Clusters using Mantri", In Osdi, vol. 10, no. 1, pp. 24. 2010.

[28] Chen Q, C. Liu, and Z. Xiao, "Improving MapReduce performance using smart speculative execution strategy," IEEE Trans.Comput., vol. 63, no. 4, pp. 954–967, 2014.

[29] Ibrahim S, H. Jin, L. Lu, B. He, G. Antoniu, and S. Wu, "Maestro: Replica-aware map scheduling for MapReduce," in Proc. 12th IEEE/ACM Int. Symp. Cluster Cloud Grid Comput, 2012, pp. 435–442.

[30] Chen, Quan, Daqiang Zhang, Minyi Guo, Qianni Deng, and Song Guo, "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, 2010.

[31] Y. Kwon, M. Balazinska, and B. Howe, "A Study of Skew in Mapreduce Applications," Proc. Fifth Open Cirrus Summit, 2011.

[32] Kandula Ananthanarayanan S, A. Greenberg, I. Stoica, Y. Lu, B. Saha, and E. Harris, "Reining in the Outliers in Map-Reduce Clusters Using Mantri," Proc. Ninth USENIX Conf. Operating Systems Design and Implementation, (OSDI '10), 2010.

[33] Qi Chen, Cheng Liu, and Zhen Xiao, "Improving MapReduce performance using smart speculative execution strategy", IEEE Transactions on Computers, vol. 63, no. 4, pp. 954-967, 2013.

[34] Alpaben K Patel, JyotiDivecha, "Modified exponentially weighted moving average (EWMA) control chart for an analytical process data", Journal of Chemical Engineering and Materials Science, vol. 2, no. 1, pp. 12-20, 2011.

[35] https://www.wallstreetmojo.com/ewma/