A simulation-based approach for straggler tasks detection in Hadoop MapReduce

Vivek M[1], Annappa Swamy D R[2]

[1]Masters in Technology Student, Dept. of Computer Science and Engineering, Mangalore Institute of Technology and Engineering, Moodbidri.
[2]Associate Professor, Dept. of Computer Science and Engineering, Mangalore Institute of Technology and Engineering, Moodbidri.

---***---

Abstract - *Hadoop MapReduce splits an incoming job into smaller individual tasks that can execute in parallel on several servers. These servers can be combined to form massive clusters with heterogeneous system configurations. The resource manager is in charge of allocating these actual tasks among the cluster's nodes in order to finish the work. When all of these distinct tasks have finished their work, the job is considered to be accomplished. Straggler tasks are those that require an extended period of time to perform. These delayed tasks will have an impact on the total MapReduce Job completion time. Straggler tasks might emerge due to a variety of factors, including hardware, network, or configuration difficulties.*

Hadoop offers speculative execution to shorten the time it takes for a job to respond by assigning straggler tasks to other nodes in the cluster to complete them. We can't anticipate a faster task completion time in the Hadoop cluster with more speculative tasks. Due to the possibility that the node already handling the straggler task will finish it before the node that just started executing the straggler task. In that situation, the freshly launched straggler will be rejected, wasting the time and resources of that specific node. Therefore, we propose a simulation-based approach that uses machine learning to identify straggler tasks based on their expected completion time. The suggested approach has the potential to considerably increase the overall performance of Hadoop MapReduce job execution time.

Key Words: Resource Manager, Straggler task, Speculative execution, MapReduce, Machine learning

1. INTRODUCTION

Due to the growing usage of the web in everything currently, a lot of information is created and reviewed. Each user's behaviour on social media and web crawlers are tracked and analysed in order to optimise website design, fight spam and extortion, and discover new marketing opportunities. Facebook's petabyte-scale data warehouse houses the 4 petabytes of information it generates every day. This rapid expansion of information has made it more challenging to organise, analyse, interpret, utilise, and make judgement. Big Data management techniques need to be continually improved in light of these difficulties. The processing of this enormous amount of data makes use of distributed and simultaneous processing techniques. The Map-Reduce programming model is one of the most used methods. MapReduce breaks down a job into smaller jobs that run concurrently on many computers.

Hadoop's job execution is built on one fundamental idea: bringing data towards computation is more expensive than moving computation towards data. As a result, in a network-constrained scenario, Hadoop attempts to organise map tasks to complete its work near to the input data chunks. Straggler tasks degrade the overall performance of Hadoop MapReduce processes. Straggler tasks take significantly longer to complete than other tasks in the server. Hadoop employs speculative execution to minimise work reaction time by concurrently initiating straggler tasks on other nodes. Because sending out a faulty speculative execution can waste cluster assets and extend job completion time. Therefore, in order to enhance the MapReduce job completion time, we suggest the use of a machine learning approach called the k-means clustering algorithm to efficiently classify fast and slow running tasks.

2. EXISTING SYSTEM

In the Hadoop framework, straggler task speculation is based on a simple profiling method that compares the progress score of each task to the average of all the tasks that are currently executing in a node. Hadoop uses the progress score parameter to estimate how much work each job has completed. A progress score is used to assess their success; it runs from 0 to 1. The amount of work accomplished for the input data read determines a map task's progress score. A Reduce task progress score is comprised of three stages, each of which contributes one-third to the overall result. If a task has been running for at least one minute and its progress score is significantly below the category average score minus 0.2, it is deemed to be a straggler. The existing technique assumes that all data centres have the same hardware configuration. However, data centres are varied in nature, with various CPU kinds, memory capacity, and other factors. Furthermore, it assumes that all activities will continue at almost the same rate, even though for running

Reduce tasks, the copy phase moves faster than the sort phase.

3. PROPSED SYSTEM

In a shared environment, the Hadoop framework selects a user task that must be scheduled using First Come First Serve (FCFS), Capacity Scheduler (CS), or Fair Scheduler (FS). The work to be scheduled is then selected from a list of available jobs. When a job's map, reduce, or speculative task is approved, it is scheduled. Because our primary goal is to successfully detect straggler tasks, we primarily focus on speculative tasks. Speculative tasks might be either unsuccessful Map or Reduce tasks. As a result, we focus mostly on speculative tasks and group them using k-means clustering.

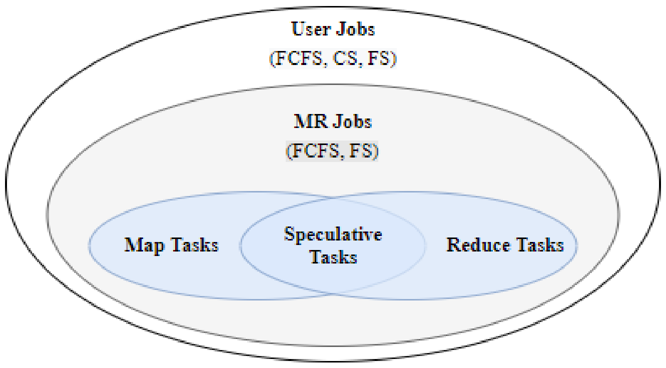
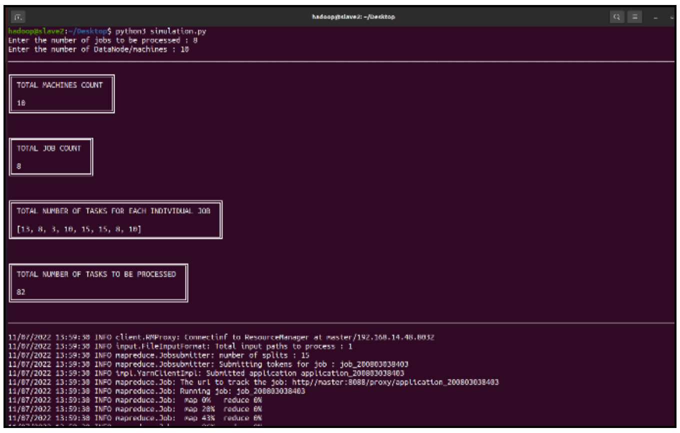


Fig -2: Scheduling schemes

K-means clustering employs "centroids," or K distinct randomly generated points in the data, and assigns each data point to the nearest centroid. We use K or centroids as 2 in our case because we only need to classify two groups. The centroid is shifted to the average of all the points allocated to it once each point has been assigned. The procedure is then repeated: each point is allocated to its nearest centroid, and centroids are relocated to the average of the points given to them. When no point alters its assigned centroid, the process completes. We can effectively classify the fast and slow running tasks in the nodes using this proposed method.

4. IMPLEMENTATION AND RESULTS

Speculative Execution is a Hadoop action that happens while a task is being carried out more slowly in a node. The Name Node begins performing the task in another node during this action, and the task that is done first is accepted, thus the execution of the other is terminated by killing that. Because in certain circumstances a Hadoop cluster with hundreds of nodes is used to process the jobs, speculative execution is advantageous. The performance of the MapReduce task will be impacted by the failure in hardware and network congestions, which frequently occur during the parallel execution of the operations. The framework groups the jobs with greater and lower completion times using k-means clustering. The software receives input in the form of the number of jobs and machines, and then randomly distributes work across N jobs. The completion of the assigned task will be determined with the time it took the machine to finish it after a certain interval when tasks T0 through T81 are completed in 18 nodes according to the clustering method.


```
hadoop@slave2:~/Desktop$ python3 simulation.py
Enter the number of jobs to be processed : 8
Enter the number of DataNode/machines : 18

TOTAL MACHINES COUNT
18

TOTAL JOB COUNT
8

TOTAL NUMBER OF TASKS FOR EACH INDIVIDUAL JOB
[13, 8, 3, 10, 15, 15, 8, 10]

TOTAL NUMBER OF TASKS TO BE PROCESSED
82

11/07/2022 13:59:38 INFO client.RMProxy: Connectinf to ResourceManager at master/192.168.14.48.8032
11/07/2022 13:59:38 INFO input.FileInputFormat: Total input paths to process : 1
11/07/2022 13:59:38 INFO mapreduce.Jobsubmitter: number of splits : 15
11/07/2022 13:59:38 INFO mapreduce.Jobsubmitter: Submitting tokens for job : job_208803038403
11/07/2022 13:59:38 INFO impl.YarnClientImpl: Submitted application application_208803038403
11/07/2022 13:59:38 INFO mapreduce.Job: The url to track the job: http//master:8088/proxy/application_208803038403
11/07/2022 13:59:38 INFO mapreduce.Job: Running job: job_208803038403
11/07/2022 13:59:38 INFO mapreduce.Job:  map 0%   reduce 0%
11/07/2022 13:59:38 INFO mapreduce.Job:  map 28%  reduce 0%
11/07/2022 13:59:38 INFO mapreduce.Job:  map 43%  reduce 0%
```

Fig -3: Screenshot of terminal window when assigning the jobs and machine count

We have given the input to the simulation, that is eight jobs for each job random tasks are allocated and simulated. For the all the 82 random tasks with their associated completion time plotted on scatter plot using matplotlib python library is scattered as data points, with task completion time on x-axis and task id on y-axis.

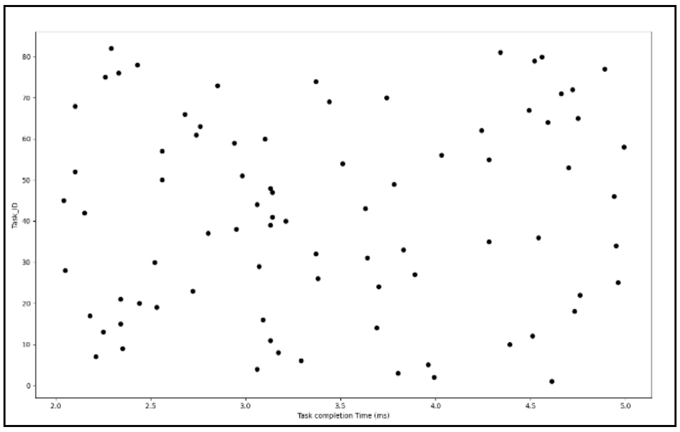


Fig -4: Scatter plot for all running task's

To determine the precise centroids for two clusters, one with a faster completion time than the other, we create a random centroid to the scatter plot. Based on the completion rate and overall task count, the mean values of the different tasks are determined. The threshold values assist in identifying tasks that are completed more quickly and at a lower rate. The Euclidean distance formula, which takes into account the point value and centroids of the task done, is used to determine the threshold value. The resulting value is then used to separate the nodes.

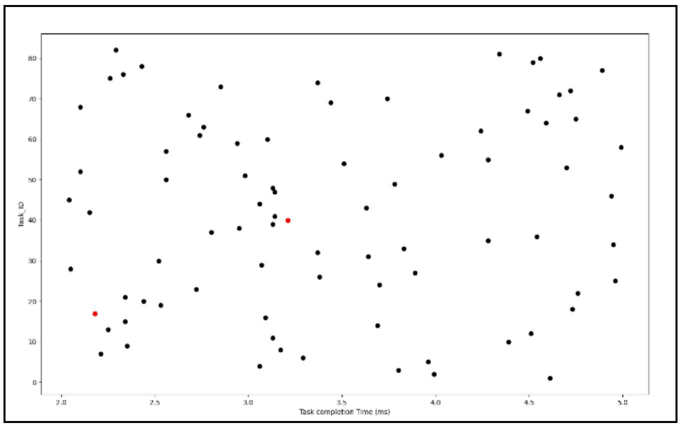


Fig -5: Scatter plot with 2 random centroids

The k instances determine how the clusters are constructed. The mean value of the items in a cluster, which may be represented as the cluster mean, is used to evaluate cluster similarity. The main goal of the framework is to find the lagging tasks in a job and back them up to other nodes to speed up the job's total response time. The K-Means method may be used to group jobs that are running more quickly and more slowly in relation to time intervals. The determination of the slow-moving tasks in the nodes occurs after the grouping of jobs has been finished. Tasks need to be moved to the machine that runs more quickly. After calculating the mean value for the job and the centroid distance of the points, the Euclidean distance is determined for the purpose of grouping. Assigning 82 tasks to 18 computers during processing allows for the detection of stagnant machines. After a period of time, the task execution time is recorded.

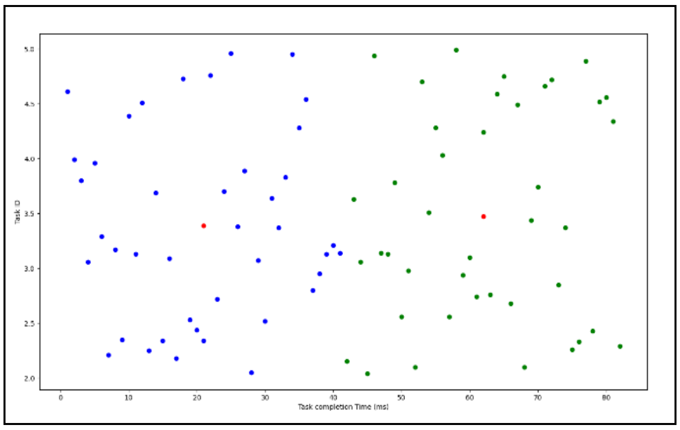


Fig -6: Scatter plot with two cluster

We can observe that there are two cluster groups, one for fast-moving jobs and another for slow-moving tasks, from the scatter plot shown above. The fastest-running tasks in the cluster may be identified by the blue colour of the data points. The tasks in the cluster that are operating slowly are those with a green colour. The two red data points represent the cluster's two centroids. We also plot a bar graph for all running tasks, fast performing tasks and slow performing tasks. Figure 7 shows a bar graph with all of the expected job task IDs plotted on the X and Y axes, respectively.

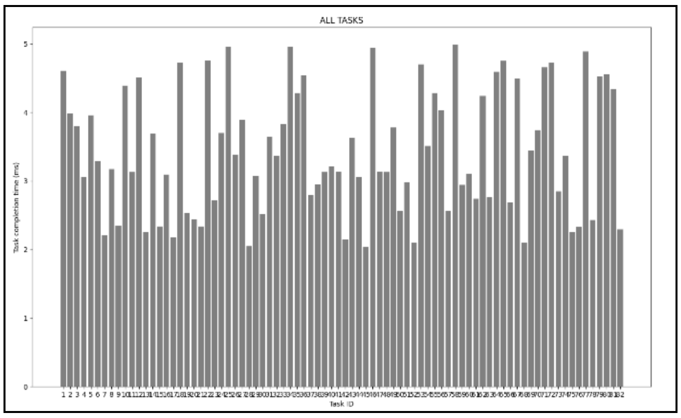


Fig -7: Bar graph for all running tasks

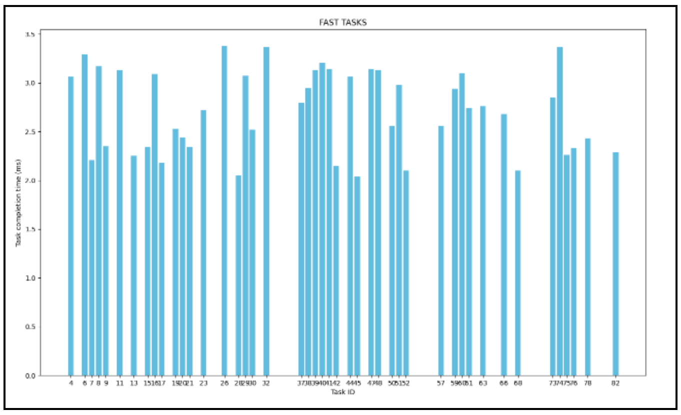


Fig -8: Bar graph for fast performing tasks

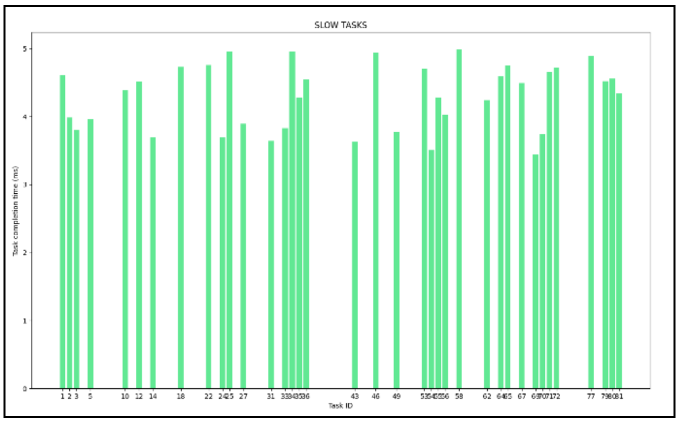


Fig -9: Bar graph for slow performing tasks

5. CONCLUSION

Hadoop, a freely available framework for processing, storing, and monitoring data, has gained a lot of development traction because it provides a distributed file system and efficient processing for large amounts of data using the MapReduce programming style. The Hadoop cluster will be able to execute speculation more effectively with the assistance of accurate detection of straggler tasks using the suggested k-means clustering approach.

REFERENCES

[1] G. Ananthanarayanan, A. Ghodsi, S. Shenker and I. Stoica, "Effective straggler mitigation: Attack of the clones", in Proceedings of the 10th USENIX conference on Networked Systems Design and Implementation, pp. 185-198, Lombard, IL, USA, April 2013.

[2] M. Zaharia, A. Konwinski, A. D. Joseph, R. Katz and I. Stoica, "Improving MapReduce performance in heterogeneous environments", in Proceedings of the 8th USENIX conference on Operating Systems Design and Implementation, pp. 29-42, San Diego, CA, USA, December 2008.

[3] Q. Chen, C. Liu and Z. Xiao, "Improving MapReduce performance using smart speculative execution strategy", in IEEE Transactions on Computers, Vol. 63, Issue 4, pp. 954-967, April 2014.

[4] J. Xie, S. Yin, X. Ruan, Z. Ding, Y. Tian, J. Majors, A. Manzanares, and X. Quin, "Improving MapReduce performance through data placement in heterogeneous Hadoop clusters," in IPDPS Workshops, pp. 1–9, 2010.

[5] N. J. Yadwadkar, G. Ananthanarayanan, and R. Katz, "Wrangler: Predictable and faster jobs using fewer resources," Proceedings of ACM Symposium on Cloud Computing (SoCC), pp. 1–14, 2014.

[6] Zhenhua Guo, Geoffrey Fox, "Improving MapReduce Performance in Heterogeneous Network Environments and Resource Utilization," in Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pp 714-716, 2012.

[7] R. Buyya, C. Yeo, S. Venugopal, J. Broberg, and I. Brandic,"Cloud computing and emerging IT platforms: vision, hype,and reality for delivering computing as the 5th utility,"FutureGeneration Computer Systems, vol. 25, no. 6, pp. 599–616,2009.

[8] A. Pavlo, et al. "A comparison of approaches to large-scaledata analysis," inProc of SIGMOD '09. New York, NY,USA: ACM, 2009, pp. 165–178.

[9] O. Gotoh, "An improved algorithm for matching biologicalsequences."Journal of molecular biology, vol. 162, no. 3,pp. 705–708, Dec. 1982.

[10] H. Li, et al. "Design patterns for scientific applicationsin DryadLINQ CTP," inProc of DataCloud '11, ser.DataCloud-SC '11. ACM, 2011, pp. 61–70

[11] "Hadoop Tutorial: All you need to know about Hadoop!" history and basic concepts of Hadoop are discussed with the help of the page https://www.edureka.co/blog/hadoop-tutorial/

[12] "Setting-up Fully Distributed Hadoop Cluster" to setup 1 mater and 7 slave nodes in a cluster https://medium.com/@zaman.nuces/setting-up-fully-distributed-hadoop-cluster-series

[13] ZhuoTang, Lingang Jiang, Junqing Zhou, Kenli Li and Kenli Li "self-adaptive scheduling algorithm for Reduce start time", in Future Generation Computer Systems, pp. 51-60, February 2015.

[14] C. Lin, W. Guo, and C. Lin, "Self-Learning MapReduce Scheduler in Multi-job Environment," Proceedings - 2013 International Conference on Cloud Computing and Big Data, CLOUDCOM-ASIA, pp. 610–612, 2013.

[15] Sun, C. He, and Y. Lu, "ESAMR: An Enhanced Self Adaptive MapReduce Scheduling Algorithm," in Proceedings of the 2012 IEEE 18th International Conference on Parallel and Distributed Systems, series ICPADS'12. IEEE Computer Society, Washington, DC, USA, pp. 148-155, 2012.

[16] P. Elespuru, S. Shakya, and S. Mishra, "MapReduce system over heterogeneous mobile devices," inProceedings of the7th IFIP WG 10.2 International Workshop on Software Tech-nologies for Embedded and Ubiquitous Systems, pp. 168–179, Springer, 2009

BIOGRAPHIES

VIVEK M
Masters in Technology Student, Dept. of Computer Science and Engineering, MITE

MR. ANNAPPA SWAMY D. R.
Associate Professor, Dept. of Computer Science and Engineering, MITE