A Machine Learning Approach For Improved Scheduling In Heterogeneous Hadoop Cluster

Prof. Aihtesham N. Kazi[1], Dr. D.N.Chaudhari[2]
[1]Assistant Professor, Dept of Computer Science & Engineering
[2]Professor,Dept of Computer Science & Engineering
[1, 2] Jawaharlal Darda institute of Engineering &Technology, Yavatmal

Abstract- *Today's most popular computer applications and Internet services to millions of users produce large volume of data at server. Such large volume of data services work with interest in parallel processing of commodity clusters. A big data processing network such as Map Reduce implemented big cluster of data to compute the large amounts of raw data, such as documents, Web request logs, etc., to compute various kinds of derived data, such as inverted indices. A Map Reduce job generally splits the input into multiple inputs which are processed by the map tasks is in completely parallel manner. In this scenario it is equally importantly, if a node is available but it takes more time to perform particular task it called as straggler. The problems of straggler occur in the way when processing big data, the straggler occur in the cluster will delay the job execution time and reduce the cluster throughput the number of jobs completed per second in the cluster.*

I. INTRODUCTION

Drastically increments in volume of data leads to putting lot of burden on servers in terms of storage cost and performance issues, here we find ma scope to get the feasible solutions, like scheduling of jobs in big data framework such as MapReduce where we can discover straggler nodes by machine leaning technology to serve valuable input to scheduler so that early scheduling decision is to be made and unnecessary resource utilization can be avoided This research is an aim at designing and implementation of machine learning driven a novel straggler detection algorithms in order to improve the performance and throughput of cluster. The efficient use of the available computing devices is an important issue for heterogeneous cluster computing systems. The ability to choose a CPU or GPU processor and other resources for a specific task has a positive impact on the performance of systems. It helps to reduce the total processing time and to achieve the uniform system utilization. As a consequence, the proper job management system should be adapted to deal with such complexity by determining efficient allocation of resources and scheduling strategies that can deal with such complexity, we can adapt evolutionary strategic based on machine learning technology to solve this difficult challenges.

II. REVIEW OF LITERATURE

Study has been carried out by the author that stated that by engaging state of art machine learning techniques can speed up the optimization process which could otherwise take many machine to explore the runtime and efficiency of jobs. The complex machine learning methodologies to applied to solve difficult real world problems using parallel programming models. Parallel machine learning aims to capitalize on the simultaneous execution of sophisticated machine learning algorithms using modern hardware architectures. [3]

Recent research done by Yigitbasi et al. [8] have focused on employing machine learning-based auto-tuning for diverse MapReduce applications and cluster configurations in Hadoop framework. Work shows that HPC can be studied under two categories such as adaptation of machine learning techniques and HPC techniques for code optimization in big data frameworks. Given the problem of scheduling possible parallel tasks, that are dependent on one another, in distributed and heterogeneous systems, there are many researches and experiments published, some of them using heuristic approaches while others using evolutionary algorithms.

In this research MateiZaharia, AndyKonwinski [6] author discuss about Hadoop scheduler behavior that shows Low response time is critical for data intensive application. Hadoop's performance is closely tied to its task scheduler, that implicitly assumes that cluster nodes are homogeneous and tasks make progress linearly, and uses these assumptions to decide when to speculatively re-execute tasks that appear to be stragglers. The sheer volume of data that these services work with has led to interest in parallel processing on commodity clusters. A key benefit of MapReduce is that it automatically handles failures, hiding the complexity of fault-tolerance from the programmer. If a node crashes, MapReduce re-runs its tasks on a different machine. Equally importantly, if a node is available but is performing poorly, a condition that we call a straggler, MapReduce runs a speculative copy of its task (also called a "backup task") on another machine to finish the computation faster. Without this mechanism of speculative

execution, a job would be as slow as the misbehaving task. Stragglers can arise for many reasons, including faulty hardware and misconfiguration. Hadoop's scheduler starts speculative tasks based on a simple heuristic comparing each task's progress to the average progress, although this heuristic works well in homogeneous environments where stragglers are obvious. The speculative execution to be a simple matter of duplicating tasks that are sufficiently slow. In reality, it is a complex issue for several reasons. First, speculative tasks are not free - they compete for certain resources, such as the network, with other running tasks. Second, choosing the node to run a speculative task on is as important as choosing the task. Third, in a heterogeneous environment, it may be difficult to distinguish between nodes that are slightly slower than the mean and stragglers. Finally, stragglers should be identified as early as possible to reduce response times.

Dolly: [9] Proposed by Ganesh A. et. al, it uses the progress rate for identification of straggler. This progress rate is calculated for each phase i.e. map, reduce and join. The progress rate defines as size of its input data divided by its duration. They use progress rate instated of the task duration which remain agnostic to skews in work assignment among tasks. The advantage of this it identifies the straggler proactively. Dolly is presented in [7], which uses task cloning to address the straggler problem for small parallel processing jobs generated by interactive data analyses. Dolly launches multiple clones of every task, and only uses the output of the clone that completes first. Dolly employs a technique called "delay assignment" to avoid contention for intermediate data, which is the main challenge faced by task cloning.

Ganesh A et. al [14] proposed a Mantri, outliers detection mechanism by which identifies the point at which task are unable to make progress at the average rate. It diagnoses the straggler depending on expected execution time of the task. Which is the addition of time taken by the task till now and remaining time? This uses the progress score [20] and progress rate (used by the LATE) to find the remaining time. It categories the root cause of straggler as static and dynamic. Where the static such as hardware reliability and dynamic such as contention for processor, memory etc. It uses the network characteristics which define the data transfer rate in the system and data skew in input size of the task for straggler detection. They consider the straggler if its expected execution time is greater than 1.5 * average execution time of the task. Mantri is presented in [5], which is a system that can monitor the performance of processing nodes in MapReduce clusters and remove stragglers based on their causes. Mantri employs three major techniques: restarting tasks running on stragglers, network-aware task placement, and protecting the output of valuable tasks.

[13] Present the default straggler identification technique which is used in hadoop is based on progress score which varies in between 0 to 1. This progress score shows the amount of task executed out of the complete task. This progress score applicable for the map as well as reduce function of Hadoop

LATE (Longest Approximation Time to End Scheduling) Algorithm M. Zaharia [20] claims that progress score alone does not gives the accurate result because accurately shows how fast a task runs as different tasks start at different moments. They calculated the progress rate for each task using formula, PR[i]= PR[i]/T, where T is the amount of time the task has been running for, and then calculate the time required for the completion of task as (1 – ProgressScore) / ProgressRate. LATE performance is evaluated in two environments: large clusters on EC2, and a local virtualized tested. The Longest Approximate Time to End (LATE), an improved task scheduling algorithm for Hadoop, is proposed in [8]. LATE is developed to address Hadoop's problem of performance degradation in heterogeneous environments by speculatively executing tasks that hurt the job completion time the most based on projected task finish times. The evaluation results show that LATE can improve the job completion time of Hadoop by a factor of two. Load imbalance among cluster nodes is also a major reason for the occurrence of stragglers in parallel processing. In our previous work, we address the load balance issue of Hadoop from two different perspectives: task assignment and replica placement

Wrangler: NeerajaYadwadkaret. al. [12] proposed a proactive straggler identification using the machine learning approach. It selects the feature for ML algorithm as specified ganglia [22].It uses the support vector machine for the classification. Wrangler predicts the straggler using a linear modeling technique based on cluster resource usage counter and uses this prediction to provide the insights to the scheduler. As cause behind the straggler are varies from node to node and time to time, but wrangler is capable of adapting such a situations that complement to the straggler.

SAMR: [15] Self-Adaptive MapReduce proposed by Q. Chen which calculates the progress of the tasks dynamically and it has implemented the concept of LATE scheduling algorithm which identifies slow tasks by approximating execution time of a task. To get more accurate progress score than LATE, SAMR uses the historical information recorded on each node in the cluster to tune the weights of map and reduce stages and also it updates the weights after each task execution. Therefore, SAMR scheduler performance is enhanced in heterogeneous environment as compared to MapReduce default scheduler and LATE

scheduler. The major hurdle in Self-Adaptive MapReduce is, it does not consider that different job types can have different weights for map and reduce stages.

Qi Chen, Cheng Liu, has been stated in his paper MapReduce is a widely used parallel computing framework for large scale data processing. The two major performance metrics in MapReduce are job execution time and cluster throughput. They can be seriously impacted by straggler machines – machines on which tasks take an unusually long time to finish. Speculative execution is a common approach for dealing with the straggler problem by simply backing up those slow running tasks on alternative machines. These researches provide an analysis of the pitfalls of current speculative execution strategies in MapReduce. We present scenarios which affect the performance of those strategies: data skew, tasks that start asynchronously, improper configuration of phase percentage and abrupt resource competitions. Based on the analysis, we develop a new speculative execution strategy called MCP to handle these scenarios. MCP takes the cost performance of cluster computing resources into account, aiming at not only decreasing the job execution time but also improving the cluster throughput .[7] Maximum Cost Performance (MCP), a new speculative execution strategy, is presented in [6]. MCP uses both progress rate and process bandwidth within a phase to identify stragglers, uses exponentially weighted moving average to predict the processing speed of cluster nodes and calculate the remaining running time of tasks, and performs speculative execution based on a cost-benefit model.

Ashwin Bhandari, Jitin George shows in his paper that Distributed processing frameworks split a data intensive computation job into multiple smaller tasks, which are then executed in parallel on commodity clusters to achieve faster job completion. A natural consequence of such a parallel execution model is that slow running tasks, commonly called stragglers potentially delay overall job completion. Straggler tasks continue to be a major hurdle in achieving faster completion of data intensive applications running on modern data-processing frameworks. They reviewed and analyzed various straggler mitigation approaches. These studies have been shown the broad subcategories of straggler mitigation i.e. proactive and reactive methods.

III. ABOUT WORKING OF HADOOP

Hadoop's implementation of MapReduce closely resembles Google's [1]. There is a single master managing a number of slaves. The input file, which resides on a distributed filesystem throughout the cluster, is split into even-sized chunks replicated for fault-tolerance. Hadoop divides each MapReduce job into a set of tasks. Each chunk of input is first processed by a map task, which outputs a list of key-value pairs generated by a user-defined map function. Map outputs are split into buckets based on key. When all maps have finished, reduce tasks apply a reduce function to the list of map outputs with each key.a virtual Hadoop cluster powered by campus clouds becomes a common environment to run MapReduce jobs, which leads to non-trivial heterogeneity in resource capacities by reasons like multiple generations of hardware, the employment of edge resources, layered network connections and even different policies on resource sharing. Such heterogeneity damages the MapReduce jobs' response time by intensifying the *straggler problem* In the design of Hadoop, there is an inherent assumption that the underlying infrastructure should be homogeneous, which means that the capacities of underlying computing and network resources should be equivalent.

Assumptions in Hadoop's Scheduler

Hadoop's scheduler makes several implicit assumptions:

1. Nodes can perform work at roughly the same rate.
2. Tasks progress at a constant rate throughout time.
3. There is no cost to launching a speculative task on a node that would otherwise have an idle slot.
4. A task's progress score is representative of fraction of its total work that it has done. Specifically, in a reduce task, the copy, sort and reduce phases each take about 1/3 of the total time.

Slow tasks may dominate the life of each phase, thus dominate the response time of a MapReduce job. A slow task is called straggler task and on which it runs is called straggler. Response time is most important for short jobs that the quick answers are required, which is a major use case for MapReduce[4]. Therefore, minimizing the response time, i.e. minimizing the running time of straggler tasks is a crucial target for a MapReduce job scheduler [9].

The straggler problem can break into three sub-problems:

• How to reduce the occurrences of straggler tasks?
• How to identify a straggler task?
• How to mitigate the impact of straggler tasks?

IV. PROBLEM ANALYSIS & PROPOSED WORK

Today's computing has achieved widespread adoption due to its ability to automatically parallelize a job into multiple short tasks, and transparently deal with the challenge of executing these tasks in a distributed setting. One

such fundamental challenge is straggling tasks, which is faced by all cloud frameworks, such as MapReduce [1], Dryad [2], and Spark [3].

The MapReduce architecture provides self-managed parallelization with fault tolerance for large-scale data processing. Stragglers, the tasks running slower than other tasks of job, could potentially degrade the overall cluster performance by increasing the job completion time. The original MapReduce paper [1] identified that Stragglers could arise due to various reasons including software mis-configurations, hardware degradation, overloaded nodes or resource contention. The Hadoop use commodity hardware and the task failure become part of the straggler.

Straggler Problem:

Stragglers are tasks that run much slower than other tasks, and since a job finishes only when its last task finishes, stragglers delay job completion. Stragglers especially affect small jobs, i.e., jobs that consist of a few tasks. Such jobs typically get to run all their tasks at once. Therefore, even if a single task is slow, i.e., straggle, the whole job is significantly delayed

The Straggle in the Distributed system when the master allocate the task to slave machine to complete it , if the slave machine takes too much time to complete the particular task beyond the threshold limit so we can say the straggler occur in the particular slave machine. In the production clusters at Facebook and Microsoft Bing, even after applying state-of-the-art straggler mitigation techniques, these latency sensitive jobs have stragglers that are on average 8 times slower than the median task in that job. Such stragglers increase the average job duration by 47%.

The Straggler defined as:

Let normalized durations, $nd(ti)$ = Task execution time
Amount of work done by task *ti*
A straggler is defined if for task *ti* of a job J
$nd(ti) > (\beta \times median\{nd(ti)\})$

Where, β is threshold coefficient ($\beta \sim 1.3$) or signifies the extent to which a task is allowed to slow down before it is called a straggler. A current study shows that the straggler existing mitigation techniques, the impact of straggler in straggler tasks can be 6- 8% slower than the median task in job on a production cluster. The production is the cluster where the actually job is the job is execute in the system execute. The straggler occur 22% to 28% in the system so to eliminate or mitigate the straggler is our main objective.

Causes of Stragglers:

We can categorize the causes for stragglers into internal and external reasons as shown in table 1. Internal reasons can be solved by the MapReduce service provider, while external reasons cannot. For example, MapReduce clusters in the real world may be over-committed with multiple tasks running on the same worker node. This creates resource competition and may lead to heterogeneous performance. We can avoid this "internal reason" by limiting each worker node to run at most one task simultaneously or by only allowing tasks with different resource usage intensity to share the same worker node.

Internal Factors:

1. Heterogeneous resource capacity of worker nodes
2. Resource competition due to other MapReduce tasks running on the same worker node

External Factors:

1. Resource competition due to co-hosted applications.
2. Remote input or output source being too slow.
3. Input data skew
4. Faulty hardware

V. STRAGGLER MITIGATION TECHNIQUE

There are two way to mitigate the Straggler they are Reactive Straggler mitigation and proactive straggler mitigation.In reactive straggler mitigation when the straggler occur in the slave machine then the master run the duplicate copy to another slave machine to removing the straggler because of this throughput is degrade and job completion time is increase.

I. Reactive Straggler mitigation Technique: The problem of stragglers has received considerable attention already, with a slew of straggler mitigation techniques [1, 4, 5] being developed. These techniques can be broadly divided into two classes: black-listing and speculative execution.

1. Blacklisting: identifies machines in bad health (e.g., due to faulty disks) and avoids scheduling tasks on them. The Facebook and Bing clusters, in fact, blacklist roughly 10% of their machines. Hadoop provides manual way of blacklisting a node (by modifying the mapred-site.xml configuration file) manual blacklisting could result in wastage of resources. Other challenges to effective blacklisting are offered by Complex interactions involving network interactions, resource contentions. Black-listing is hence considered inefficient as

simple counting-based techniques or heuristics are incapable of finding the exact reasons behind slower task-executions.

2. Speculative execution: [5, 7] was explored to deal with stragglers. Instead of fixing the stragglers, Hadoop tries to detect such tasks and launches back-up copies. This is called speculative execution. Speculative execution is an optimization with a hope that these copies will finish faster. Speculative execution increases contention over the available resources resulting into higher latencies for new tasks.

3. LATE: allows the slow nodes in the cluster to be utilized as long as this does not hurt response time. In contrast, a progress rate based scheduler would always re-execute tasks from slow nodes, wasting time spent by the backup task if the original finishes faster.

4. MANTRI: Mantri where a system that monitors tasks and outliers using cause- and resource-aware techniques. These strategies include restarting outliers, network-aware placement of tasks and protecting outputs of valuable tasks. Using real-time progress reports, Mantri detects and acts on outliers early in their lifetime. Early action frees up resources that can be used by subsequent tasks and expedites the job overall.

II. Proactive straggler mitigation

In Proactive Straggler mitigation technique the machine learning technique is used to mitigate the straggler. In proactive straggler mitigation technique where the straggler occurs in slave machine predict earlier before the scheduling.

There are various way proactive technique is existing they are

1. Dolly: Clone of small jobs only marginally increases utilization the execution of the because workloads show that while the majority of jobs are small, they only consume a small fraction of the resources. The main challenge of cloning is, however, that extra clones can cause contention for intermediate data. We use a technique, delay assignment, which efficiently avoids such contention where Evaluation of our system, Dolly, using production workloads.
2. Wrangler: Wrangler is a system that predicts stragglers using an interpretable linear modeling technique. Wrangler prevents wastage of resources by removing the need for replicating tasks.Wrangler introduces a notion of a confidence measure with these predictions to overcome the modeling error problems; this confidence measure is then exploited to achieve a reliable task scheduling [7]

VI. METHODOLOGY

Today heterogeneous systems with both CPU and accelerator become more popular to adopt the demand of large-scale applications. Whether or not we can find a better scheduling policy based on the workload logs and trace file of execution of jobs from server. For this reason we mention machine learning technique and information that was extracted from these logs and traces. Fundamentally, we will apply machine learning technique called nonlinear regression to generate a dynamic scheduling function for improving the performance.

Most of the existing jobs scheduling techniques do not consider the future workload while assigning the sub-jobs to the processors [7], [12]. It reduces the resource utilization and leads to wastage of resources on jobs.

Fundamentally, the behaviors of large-scale programs on an HPC system can be extracted from the history or log files then draw a relational model between its characteristic and an objective function. This problem can be fitful in applying machine learning to find a scheduling function for job submission. Job submission with workload managers is considered as an efficient way to support users and manage the computing resources.

We use a two-stage approach: first using simulation to simulate all situations of run jobs, in order to generate a dataset which then is used for training models with machine learning.

This model enables to improve the scheduling performance and fit well the relationship between the jobs characteristics and criteria in practice.

Secondly, we evaluate all policies on the CPU/coprocessor-based cluster. This work presents the following contributions:

• We show that the possible scheduling functions can be obtained from the workload logs on heterogeneous system.
• We apply and present how to use machine learning to generate a scheduling function based on the dataset from workload logs.

VII. CONCLUSION

This study of straggler detection using machine learning approach has been done and proposed methodology is stated in this paper for implementation smart decision driven scheduler in Hadoop that can be used to improve resource utilization

REFERENCES

[1] Minh Thanh Chung, KienTrung Pham, Nam Thoai "A New Approach for Scheduling Job with The Heterogeneity-aware Resource in HPC Systems" 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems.

[2] Manu Agrawal, KartikManchanda, Akshita Agarwal, "A Supervised Approach-based Job Scheduling Technique for Distributed Real-Time Systems" , 2018 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS).

[3] Wei Dai , Ibrahim Ibrahim, Mostafa Bassiouni,"An Improved Straggler Identification Scheme for Data-Intensive Computing on Cloud Platforms", 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing.

[4] YixinBao, Yanghua Peng, ChuanWu,"Deep Learning-based Job Placement in Distributed Machine Learning Clusters" ieee transaction 2019.

[5] Ganesh Ananthanarayanan, Ali Ghodsi, Scott Shenker, Ion Stoica, "Effective Straggler Mitigation: Attack of the Clones", 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI '2013).

[6] Yashverdhan P Singh, ShyamDeshmukh ,"Improvement Of Scheduling In Hadoop Using Machine Learning", JETIR (ISSN-2349-5162) April 2017, Volume 4, Issue 04

[7] Qi Chen, Cheng Liu, and Zhen Xiao, "Improving MapReduce Performance Using Smart Speculative Execution Strategy", IEEE Transactions On Computers 2014.

[8] Ashwin Bhandare, Jitin George, Supreet Deshpande, "Review and Analysis of Straggler HandlingTechniques", (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 7 (5) , 2016,2270-2276

[9] Xue Ouyang, Peter Garraghan, David Mckee,"Straggler Detection in Parallel Computing Systems through Dynamic Threshold Calculation", 2016 IEEE 30th International Conference on Advanced Information Networking and Applications.

[10] Giacomo Domeniconi, Eun Kyung Lee, AlessandroMorari "CuSH: Cognitive ScHeduler for Heterogeneous HighPerformance Computing System". DRL4KDD, 2019, Alaska – USA.

[11] Zixia Liu, Hong Zhang, Bingbing Rao, Liqiang Wang, A Reinforcement Learning Based Resource Management Approach for Time-critical Workloads in Distributed Computing Environment.

[12] W. K. V. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer Towards Efficient Mapping, Scheduling, and Execution of HPC Applications on Platforms in Cloud Simulation Of Hpc Job Scheduling And Large-Scale Parallel Workloads, Proceedings of the 2017 Winter Simulation Conference

[13] Albert Reuther, Chansup Byun, William Arcand, Scheduler Technologies in Support of High Performance Data Analysis.

[14] Kazumasa Sakiyama,, Shinpei Kato, Yutaka Ishikawa, Atsushi Hori, Abraham Monrroy,, Deep Learning on Large-scale Muticore Clusters, 2018 30th International Symposium on Computer Architecture and High Performance Computing

[15] Eric Gaussier, David Glesser "Improving Backfilling by using Machine Learning to Predict Running Times." SC '15, November 15-20, 2015, Austin, TX, USA