BLACKLISTING BASED MACHINE LEARNING STRAGGLER IDENTIFICATION AND MITIGATION TECHNIQUE FOR DATA INTENSIVE COMPUTING IN CLOUD ENVIRONMENT

Vaibhav Pawar [1], Shyam Deshmukh. [2]

[1]Department of Information Technology, Pune Institute of Computer Technology
Savitribai Phule Pune University, Pune

[2]Department of Information Technology, Pune Institute of Computer Technology
Savitribai Phule Pune University, Pune

ABSTRACT:

The priority of data storage vendors is faster execution time and optimal resource utilization. Stragglers (slow running nodes) till date continue to be a major problem for faster job completion in distributed systems. Existing techniques however result in more job completion time and wastage of resources. A model which neglects these slow running nodes and optimizes resource utilization, providing minimum job completion time is discussed. This project aims to builds a machine learning based straggler mitigation technique using Apache spark framework which will be deployed on Cloud Environment that detects whether a node will be a straggler or not. This Machine learning based Blacklisting scheduler can detect the straggler node regardless of internal and external causes of Straggler. This project work provides an approach which can be effectively utilized in Machine Learning Systems to predict and avoid Stragglers. Various experiments were carried out using default apache spark scheduler and with machine learning based Blacklisting scheduler using input programs such as WordCount and TeraSort. The results shows that our approach reduces the Job Completion Time of task by 28% and gives better utilization of resources in cloud environment.

Keywords: Distributed Systems, Machine Learning, Straggler, Cloud Environment, Apache Spark.

[1] INTRODUCTION

The Stragglers are slow running nodes in cloud environment. In order to decide whether a node in cloud computing environment is straggler we need node configuration and we need to check how much tasks or jobs the particular node has executed in a given time. So it is necessary for us to schedule the jobs on the node to decide whether the node is straggler or not. For this purpose we need scheduling algorithms for mitigating the effect of stragglers. Stragglers are slow running nodes in cloud environment. In order to decide whether a node in clod computing environment is straggler we need node configuration and we need to check how much tasks or jobs the particular node has executed in a given time. So it is necessary for us to schedule the jobs on the node to decide whether the node is straggler or not. For this purpose we need scheduling algorithms for mitigating the effect of stragglers.

[1.1] ROOT CAUSES OF STRAGGLER:

Internal factors:

- Heterogeneity in the resource capability of a node.
- Competition between the tasks for resource allocation.

External factors:

- Competition for resources Co-hosted applications.
- Skewed Input dataset.
- Slow Processing speed of remote input/output source.
- Defective Hardware.

[1.2] MOTIVATION:

A job can finish its execution earlier as compared to that of individual processing. It would be very good if the job completion time can be further optimally reduced as compared to the existing reactive solutions . So this area of research is interesting. Google first introduced the MapReduce framework in order to parse and the enormous amount of data. Scalability of MapReduce Framework is very high with respect to large clusters. Hadoop is an popular example that shows how the map reduce framework is implemented in an open source manner and has been widely used by industries of both small and large sizes. For speeding up a job's execution time, MapReduce breaks a job into multiple smaller tasks. Execution of these tasks takes place in parallel manner and then submitted to multiple node cluster. Job is said to be completed when all the tasks related to that job has finished its execution. The major advantage of such distributed parallel processing environment is that they have automatic fault tolerance capability, without needing extra costs from the programmer.

[1.3] SCOPE:

The scope of the study is to reduce the execution time of the jobs running on the distributed systems environment. This study also tries for Stragglers avoidance, Improvement in resource utilization , Monitoring of task parameters like CPU utilization, RAM or memory usage , Network resources usage and bandwidth usage, Input/output Disk Usage, number of threads related to task.

[1.4] PROBLEM STATEMENT:

To implement improved Straggler identification and mitigation scheme in Cloud environment.

[2] LITERATURE SURVEY

Straggler nodes are the nodes on which a task takes much longer than normal to finish. Straggler nodes increase execution time and reduce cluster throughput. Straggler nodes degrade the cluster throughput. Straggler tasks are a hurdle in getting faster completion of applications on modern frameworks. There are various Machine Learning Algorithms that are also discussed in this paper such as Self-Adaptive MapReduce Scheduling Algorithm (SAMR), Multi-task Learning and Blacklisting.

[2.1] Internal Factors For Straggler Node:

Some node involved in the cluster can have resource requirement that may not available in the same environment of the cluster this gives rise the heterogenous requirements of resources in the cluster. So in order to solve this problem there exists a Longest Approximate Time to End (LATE), it will relaunch only the slowest tasks, and only a small number of tasks.

There are situations where the tasks executing within particular tasks may compete for the resources within the node. This created the race condition among the tasks within the node for the resources so in order to avoid this situation there exists a technique that will replicate the resource instances so that no two tasks will compete for same resource at a time. This process is called as Cloning of resources This cloning of resources reduces the JCT by 34%.There exists another technique to avoid the Competition of resources within the node which called as Delay Assignment which do not immediately assign the resources to the node it first checks the previous performance of a node by collecting node's history and then based on this the resources are then allocated so it avoids the contention of resources within the node and reduces the effect of a straggler node by 46%.

[2.2] External Factors For Straggler Node:

Sometimes the **Co-hosted application** within the cluster may also compete for the resources which may cause a node be a straggler node so to avoid this there exists a technique called as Speculative execution in which each slow tasks is backed up other node in order to reduce the job execution time. Sometimes there might be scenario where the master node or on the worker node in the cluster may fail due physical damage, lack of power supply or any other natural disaster. Which indeed can causes node to stop directly making it a straggler node. Frequency of occurrence of **hardware fault** is 7% and is least consider factor for the cause of straggler.

[2.3] Machine Learning Based Methods To Handle Stragglers:

The problem with Hadoop system is that since the tasks are given to different nodes, some slow nodes may delay execution of the whole program. Since tasks run in isolation, they do not know where the input is coming from and have to trust the Hadoop platform to provide the correct input to correct task. Sometimes, this results in the same input getting processed multiple times to use different machine capabilities. As most tasks are about to finish, Hadoop schedules copies of remaining tasks to many nodes which are free. This is called **speculative execution.**

Self-Adaptive MapReduce Scheduling Algorithm (SAMR) SAMR algorithm has 2 steps first step is to map task into consideration for the first time and also it classifies slow performing nodes into straggler node. SAMR makes use of historical data stored on every node in the cluster to make a more accurate prediction of progress score than LATE but it does not consider that different job types can have different weights for map and reduce stages.

Multi task learning (MTL) This formulation is mainly used to capture the structure of our tasks and reduces job completion times by up to 59%. This reduction can be achieved with the help of 7-point prediction accuracy of straggler node. MTL formulation is generally applicable for prediction problems in distributed computing frameworks, such as resource allocation.

[2.4] Blacklisting Based Method For Straggler Mitigation Technique Using Machine Learning

Blacklisting identifies machines in bad health (e.g., due to faulty disks) and avoids scheduling tasks on them. The Straggler nodes are not blacklisted for entire Job completion after the Blacklist timeout completes a node is ready to execute tasks. So other non-blacklist node can also be backlisted this takes place with the help of machine learning algorithm based on the decision tree. The decision tree helps the scheduler to makes the decision that which node to blacklist next, meanwhile other non-blacklisted nodes can carry the execution of task. Therefore this avoids the the straggler nodes to not get the task execution and only non-straggler nodes are active and executes the task reducing the Job Completion Time(JCT).

[3] IMPLEMENTATION DETAILS

The system implementation will be using Apache spark on a multi-node Master Slave cluster Architecture. There will a master node that will assign the jobs to slaves in the form of smaller tasks and the slave nodes will execute these tasks and then the result will be integrated by master node. If a straggler is found, the Scheduler will notify the master node and no job will be assigned to that slave node until a suitable slave node is available and then assign the task to that particular slave node. There will be one master node and four slave node.

[4] EXPERIMENTAL SETUP

HARDWARE:	
MASTER NODE	CPU:-Intel Core i3 Frequency:-2.0GHz RAM: 8GB, Hard Drive: 1 TB
SLAVE NODE	Four Slave nodes with CPU: Intel Core i3 Frequency: 1 GHz RAM: 4 GB, Hard Drive: 500 GB
SOFTWARE:	**VMWARE 10 OR ABOVE** for having multiple Virtual machines running
WORKLOADS	Wordcount, Pi Estimator, Tera-sort programs will be executed on each node to check the performance.
APACHE SPARK	For Cluster Configuration

[5] PROPOSED SYSTEM ARCHITECTURE

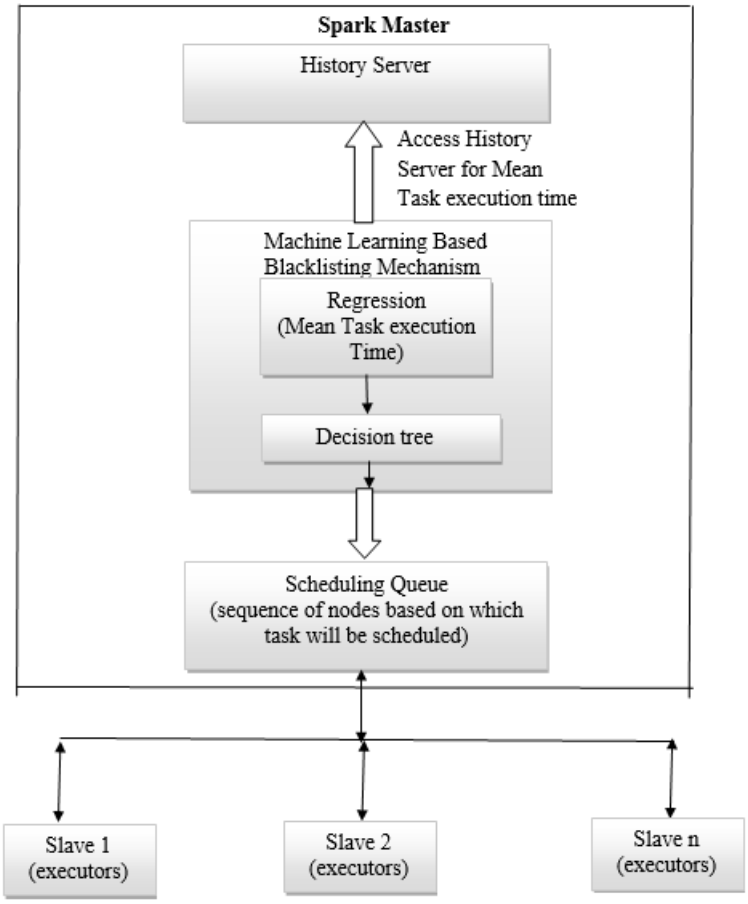

Fig.5.1 Proposed System for Straggler identification and mitigation

As shown in the fig 5.1, the execution history of WordCount and TeraSort is stored in the history server. The history server stores the execution history considering the parameters like CPU utilization of a node, Memory utilization of a node, Disk utilization, Task execution time, Location of a node. This history is given to the decision tree, based on this, the Predictive model is generated which indeed predicts the job completion time and then this is given to spark scheduler. The spark scheduler consults with Machine Learning based Blacklisting mechanism to determine which node to blacklist and in such a way straggler node is blacklisted.

[6] ALGORITHM

Let $N = (Ni : i=\mathbf{1, \dots , n}$ nodes) be the set of nodes in spark cluster.

Let N_{Bi} be the set of nodes which are blacklisted.

Let PARAM(Ni) be the set of tuples specifying the parameters that are considered to decide whether the node Ni is straggler or not.

The tuple consist of following normalized parameters :

a. CPU_USAGE
b. MEMORY_USAGE
c. DISK_USAGE
d. TASK_EXECUTION_TIME
e. LATENCY_TO_GET_THE_RESOURCE
f. STRAGGLER_FREQUENCY
g. maxTaskAttemptsPerExecutor
h. maxTaskAttemptsPerNode
i. maxFailedTasksPerExecutor
j. maxFailedTasksPerExecutor
k. maxFailedExecutorsPerNode
l. maxFailedExecutorsPerNode.

Where,

$$\text{STRAGGLER\_FREQUENCY} = \frac{\text{No. of times the node is blacklisted}}{\text{Overall Job Completion Time.}}$$

Also the vales in defaults-conf file in spark home directory are being set as per below for better results:

CONFIGURATION_VARAIBLE(Ni)= {
maxTaskAttemptsPerExecutor ,
maxTaskAttemptsPerNode ,
maxFailedTasksPerExecutor,
maxFailedTasksPerExecutor ,
maxFailedExecutorsPerNode ,
maxFailedExecutorsPerNode} =2
}

Let Sq be the Scheduling queue consisting the sequence of nodes which will be used to execute the tasks.

I) ALGORITHM: BLACKLIST_OF_NODES (Ni,Spark_Conf(Ni))

1: Spark_Conf(Ni) ← PARAM (Ni)

//Let us Construct a decision tree as per below:

MEAN_PARAM(Ni) = $\sum_{i=1}^{n} \frac{PARAM(Ni)}{Ni}$

II) ALGORITHM:

DECISION_TREE (PARAM(NI), MEAN_PARAM(Ni), CONF_VARAIBLE(Ni))

1: Let OLD_MEAN_PARAM (Ni) = MEAN_PARAM (Ni)

NEW_MEAN_PARAM = $\sum_{i=1}^{n} \frac{PARAM(Ni)}{Ni}$

2: Let OLD_CONF_VARIABLE (Ni) = CONF_VARAIBLE (Ni)

3: If (NEW_MEAN_PARAM != OLD_MEAN_PARAM (Ni)

{

$N_{Bi!}$ = Ni // N_{Bi} is Straggler-node

}

else

{

Ni→Sq[rear] //Node Ni is placed at the end of the queue

}

III) FAIR_SHARE_SCHEDULING (Sq).

DRIVER_PROGRAM(INPUT_DATA,INPUT_PROGRAM):

{

BLACKLIST_OF_NODES (Ni,Spark_Conf (Ni));

DECISION_TREE (PARAM(NI), MEAN_PARAM (Ni), CONF_VARAIBLE(Ni));

FAIR_SHARE_SCHEDULING (Sq);

Return JCT;

}

[7] RESULTS

With the help of straggler mitigation technique based on Blacklisting enabled machine Learning Approach it was observed that, Input workload(WordCount and TeraSort) gave encouraging results using spark framework It was also observed that using this approach we were able to improve the Job Completion Time approximately by **28%** on Blacklisting enabled Machine Learning scheduler for WordCount Job and Job Completion Time approximately by **19%** on Blacklisting enabled Machine Learning scheduler for Terasort job.

[7.1]Results of Default Spark Scheduler vs. Blacklisting enabled Scheduler vs. Blacklisting based ML scheduler for WordCount Job.

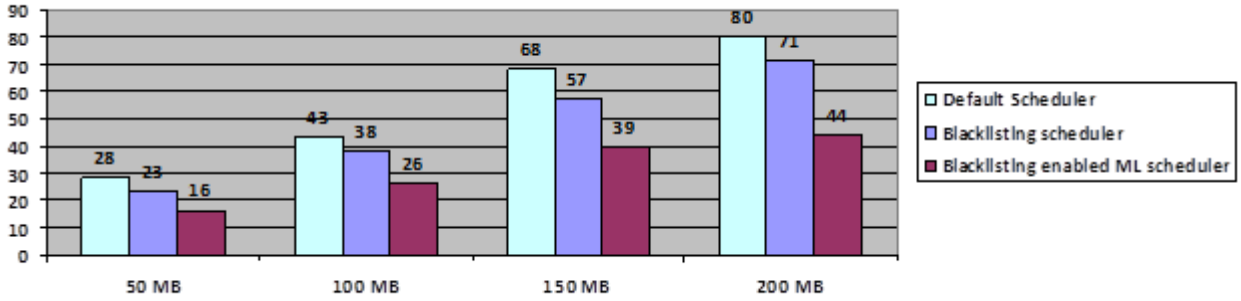

Fig 7.1 Iterations on Default Scheduler vs. Blacklist enabled Scheduler vs. Blacklisting based Machine Learning Scheduler(WORDCOUNT)

The Fig.6.1 shows that,Wordcount job execution time with Default Scheduler vs.Blacklist enabled Schedular and Blacklist enabled Machine learning scheduler. It is observed that the overall job completion time(in seconds) is reduced by **28%** on an average when Blacklisting enabled Machine learning scheduler is compared with Default spark scheduler and overall job completion time(in seconds) is reduced by **11%** on an average when Blacklisting enabled scheduler is compared with Default Scheduler . Reduction in job completion time is less when input data size is small but it rises in significance with increase in size

[7.2]Results of Default Spark Scheduler vs. Blacklisting enabled Scheduler vs. Blacklisting based ML scheduler for TeraSort Job.

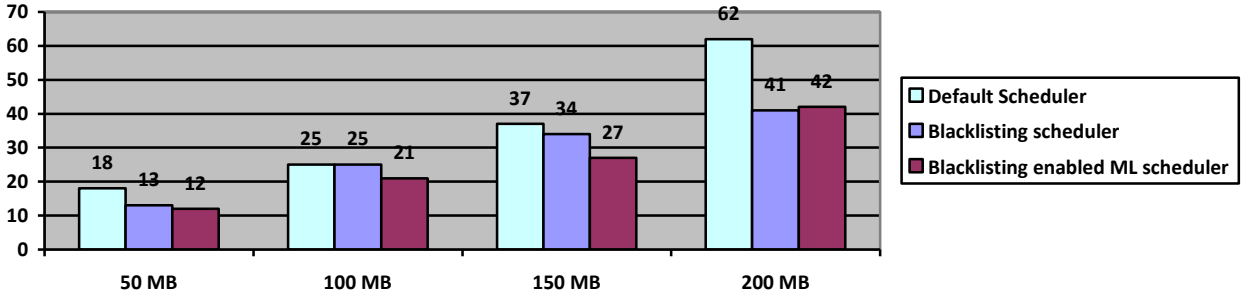

Fig 7.2 Iteration on Default Scheduler vs. Blacklist enabled Schedular vs. Blacklisting based Machine Learning Scheduler(TeraSort)

The Fig 7.2 shows that, Terasort job execution time with Default Scheduler vs.Blacklist enabled Scheduler and Blacklist enabled Machine learning scheduler. It is observed that the overall job completion time(in seconds) is reduced by **19%** on an average when Blacklisting enabled Machine learning scheduler is compared with Default spark scheduler and overall job completion time(in seconds) is reduced by **13%** on an average when Blacklisting enabled scheduler is compared with Default Scheduler . Reduction in job completion time is less when input data size is small but it rises in significance with increase in size.

[7.3] JCT vs. File Size

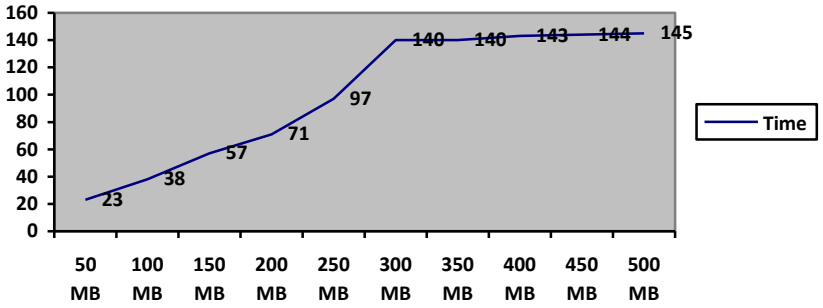

Fig 7.3. JCT vs. File Size (WordCount)

The Fig 7.3 shows the result for Word Count Job being ran with different file sizes. From the graph it is evident that after of size of approximately **500 MB** the job completion time doesn't change drastically and remains approximately constant which is approximately 140 sec hence this is the value of convergence.

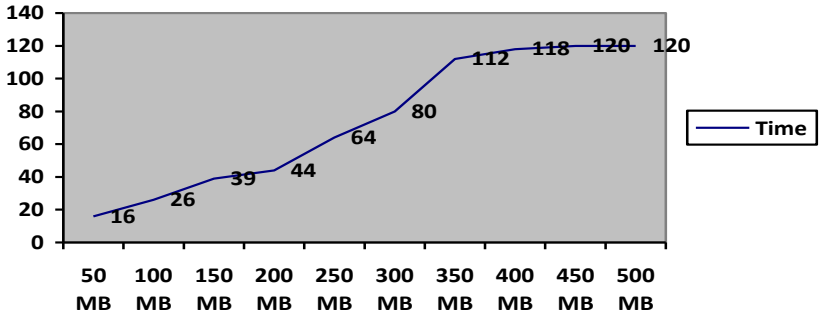

Fig 7.4. JCT vs. File Size (TeraSort)

The Fig 7.4 shows the result for TeraSort Job being ran with different file sizes. From the graph it is evident that after of size of approximately **500 MB** the job completion time doesn't change drastically and remains approximately constant which is approximately 120 sec hence this is the value of convergence.

[8] CONCLUSION

In this paper we presented various straggler mitigation approaches. We have studied the Blacklisting based Machine Learning Techniques. The Machine Learning Technique combined with Blacklisting mechanism proving to have better performance than others. We have compared all the techniques with each other based on the performance. Blacklisting enabled machine Learning Approach proved to show better results as compared to other techniques.

[9] FUTURE SCOPE

The existing techniques are meant for straggler mitigation or for reducing the impact of stragglers. In future we can have techniques for elimination of stragglers altogether without resource blacklisting and optimal use of resources. In future by considering few more parameters from spark logging mechanism and workload characteristics, the performance and the throughput of the Spark Cluster can be improved. Thus we could try to generalize the technique for all kinds of datasets eventually improving it's confidence measure too.

REFERENCES

[1] Xue Ouyang1, Peter Garraghan1, Renyu Yang2, Paul Townend1, Jie Xu1 "Reducing Late-Timing Failure at Scale: Straggler Root-Cause Analysis in Cloud Datacenters" In IEEE DSN 2016.

[2] Understanding the Dark Side of Big Data Clusters: An Analysis beyond Failures in 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks by Andrea Ros`, Lydia Y. Chen and Walter Binder.

[3] Failure Analysis of Jobs in Compute Clouds: A Google Cluster Case Study, in 2014 IEEE 25th International Symposium on Software Reliability Engineering by Xin Chen, Charng-Da Lu and Karthik Pattabiraman

[4] A Survey on Hadoop MapReduce Framework and the Data Skew Issues, International Journal of Scientific Research Engineering & Technology (IJSRET), April 2015 , by Nawab Wajid, Prof. Satish S, Prof. Manjunath T N.

[5] Dynamic Resource Allocation for MapReduce with Partitioning Skew, IEEE TRANSACTIONS ON COMPUTERS, September 2014, Zhihong Liu, Qi Zhang, Reaz Ahmed, Raouf Boutaba, Yaping Liu, and Zhenghu Gong , Members, IEEE, Xue Ouyang, Peter Garraghn, David Mckee, Paul Townend, Jie Xu "Straggler Detection in Parallel Computing Systems through Dynamic Threshold Calculation " In 2016 IEEE 30th International Conference on Advanced Information Networking and Applications.

[6] Neeraja J. Yadwadkar, Ganesh Ananthanarayanan , Randy Katz , "Wrangler: Predictable and Faster Jobs using Fewer Resources " In SoCC '14, 3-5 Nov. 2014, Seattle, Washington, USA.

[7] Matei Zaharia, Andy Konwinski, Anthony D. Joseph, Randy Katz, Ion Stoica "Improving MapReduce Performance in Heterogeneous Environments " In 8th USENIX Symposium on Operating Systems Design and Implementation.

[8] Ganesh Ananthanarayanan, Ali Ghodsi1, Scott Shenker, Ion Stoica "Effective Straggler Mitigation: Attack of the Clones " In University of California, Berkeley , KTH/Sweden.

[9] Mukhtaj Khan, Yong Jin, Maozhen Li, Yang Xiang and Changjun Jiang " Hadoop Performanc12. JaideepDhok and VasudevaVarma, Search and Information Extraction Lab,International Institute of Information Technology, Hyderabad, India "Using PatternClassification for Task Assignment in MapReduce", 10th IEEE/ ACM conference,2010.

[10] Lei Wang, RuiRen, Jianfeng Zhan, and Zhen Jia, Institute of ComputingTechnology, Chinese Academy of Sciences. "Characterization and ArchitecturalImplications of Big Data Workloads",2016 IEEE.

[11] Zhen Jia1,2, Jianfeng Zhan 1*, Lei Wang1, Rui Han1, Sally A. McKee3, QiangYang1, Chunjie Luo1, and Jingwei Li ,State Key Laboratory Computer Architecture,Institute of Computing Technology, Chinese Academy of Sciences "Characterizingand Subsetting Big Data Workloads" , 2014 IEEE.