ISSN : 2394-2975 (Online)
ISSN : 2394-6814 (Print)
Vol. 2, Issue 2 (Apr. - June 2015)
International Journal of Advanced Research in Education Technology (IJARET)

Improving Mapreduce Performance by Speculative Execution Strategy Considering Data Locality and Data Skew

[I]Suprit S H, [II]Prashanth B.S
[I]M.Tech Student, Dept of CSE, Mangalore Institute of Tech. and Engg., Mangalore, Karnataka, India
[II]Asst. Professor, Dept of CSE, Mangalore Institute of Tech. and Engg., Mangalore, Karnataka, India

Abstract
This paper presents the study of MapReduce performance in Hadoop architecture, the job assigned to the Hadoop is divided in to tasks among the node of the cluster. But some of the nodes may be running slowly in the cluster to result in slow processing of the MapReduce because of their processing capability of load on the node. Such tasks that are assigned to the slow running node in the cluster are efficiently backed up on some other nodes. There are some existing strategies to backup the slow running tasks on the alternate machines. This paper emphasizes considering data locality and data skew. The backup task should be allocated to the nodes that are the data local. Some of the data blocks vary in block sizes that is skew in the size of the data.

Keywords
MapReduce, Backup, Data locality, Data Skew, Straggler machiness

I. Introduction

MAPREDUCE [1] is proposed by Google in 2004 and has become a popular parallel computing framework for big data processing. In a typical MapReduce job the masters (job tracker) divides the input files into number of map and reduce tasks, and then allocates both map and reduce tasks to slave nodes in the cluster for processing the tasks in parallel. When a slave node takes longer time to process a task (the node will be called straggler machine), it will delay the time of execution of the job significantly. This problem is discussed and analyzed with a speculative strategy for the execution delay in this paper. Speculative execution is running the backup tasks of the slow running nodes on alternative machines with the hope of efficiency in the performance.

Hadoop [2] is an open-source implementation of MapReduce. The speculative execution study provides previously existing study on it, the original speculative execution strategy is used in Hadoop-0.20 (called Hadoop-Original), and it simply identifies a task as slow running when the progress of the task is slower than all the tasks by a certain threshold. The Hadoop-original has some drawbacks, so a new strategy called LATE [4] was proposed which is implemented in Hadoop-0.21 with little of modifications. The HADOOP-LATE keeps the progress rate (progress/time) of tasks and estimates their remaining time (progress remaining/progress rate). This strategy will select a slow running task with the longest remaining time, when there are only few map or reduce tasks left in the processing.

Then a new strategy called Microsoft Dryad [3] was proposed to support MapReduce. After this Mantri proposed a new speculative strategy for Dryad. The significant difference between LATE and Mantri [5] is that Mantri uses the task's process bandwidth (processed data / time) to calculate its task's remaining time (data left / process bandwidth).

All the strategies mentioned above have some of the drawbacks in processing the tasks and identifying the slow tasks or selecting the backup worker nodes. The strategies uses average process speed (progress rate or process bandwidth) of task to compute the remaining time, that assumes that a task progress at a stable rate. But this can be false due to many reasons. First is in MapReduce, a task is divided into multiple phases with affixed ratio of progress for each, but those phases tend to vary in different jobs and application environments. This leads to progress rate fluctuation across different phases. Second reason is reduce tasks can be launched asynchronously before completion of map tasks. In this case the progress rate of the reduce tasks will be faster at the beginning as some of the map tasks would have been completed processing and map outputs have been ready. After copying those ready map outputs, the progress rate of the reduce tasks will slow down.

The existing strategies has drawback on managing data locality. The tasks that are assigned to the slave node might not be having the data required for it to process, so it may need to request the data from different data nodes to process the task which will take more time to process. Then one more is data skew, data skew means variation in the data size, that is some data blocks can be of varying sizes (not of the same size as specifies by the block size of the MapReduce) that needs to be processed efficiently.

MapReduce and its open source implementation Hadoop have made large scale data analysis widely accessible [8]. Such runtime systems free users from problems associated with distributed coordination, fault-tolerance, and scalability. However, users may still suffer from performance problems related to skew if they are not careful regarding their map and reduce implementations and how data is partitioned across tasks.

In this paper, we propose an improvement to these Maximum Cost Performance (MCP) to address the drawbacks successfully: i) Use both process bandwidth and progress rate within a phase to choose slow running tasks in the cluster, ii) it uses exponentially weighted moving average (EWMA) to predict process progress and task's remaining time, iii) determining the tasks to backup based on the cost benefit model, iv) it also includes data locality concerned issues.

In summary, we make the following contributions. It explains some of the strategies that are already been implemented that are having some drawbacks. The further section contains the improvement to the issues. It contains the implementation module, explanation to the changes made to the existing module, then result and conclusion part explains the papers motive.

II. Literature Survey

In this section, we study an explanation for MapReduce mechanism and architecture, an overview of the reasons of stragglers, and then describe the inner mechanisms of some widely used speculative execution strategies.

ISSN : 2394-2975 (Online)
ISSN : 2394-6814 (Print)

A. MapReduce Mechanisms

In MapReduce Hadoop cluster, when a job is submitted, the master divides the input files into multiple map and reduce tasks to distribute among the slave nodes. All the worker nodes runs the assigned tasks on their available task slots and keeps updating about the progress of the tasks to the master by a periodic heartbeat. Map tasks retrieves key-value pairs from input file. Then transfer the key, value pairs to some user defined map function and combine function, and finally generate the intermediate map outputs. After this, the map outputs are made as inputs to reduce, and merge those outputs to a single order (key, value) pair stream. Then the result is generated by transferring the output to a user defined reduce function.

The map tasks are divided into map and combine phases, whereas reduce tasks are divided into copy, sort and combine phases. Reduce tasks can start only after some of the map tasks are complete but no reduce task can start sort phase until all the map outputs are available for reduce phase.

The flow diagram for the MapReduce mechanism is shown in the following figure-1.

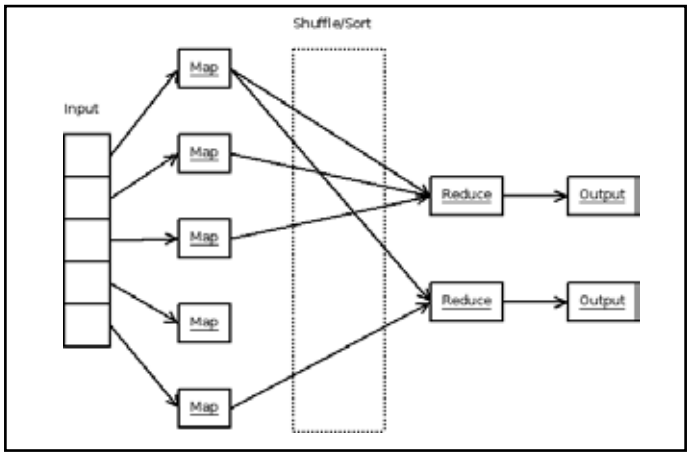

Fig 1: Flow of MapReduce mechanism.

B. MapReduce Architecture

In the MapReduce architecture first a client submits job to job tracker in the master node. Then the job tracker communicates with name node and creates execution plan. The job tracker will submit the tasks to the task tracker. Each task tracker will execute the tasks allocated to the available slots in the data nodes. Then each task tracker in the slave nodes will report the progress of the tasks via periodic heartbeat updates. Job tracker manages different phases in the MapReduce mechanism and it updates the status.

C. Causes of Straggler

The causes of straggler can be classified into internal and external reasons as shown in Table 1.

Table 1: Causes of Straggler

Internal factors	External factors
• Heterogeneous capacity of slave nodes • MapReduce tasks running on the same slave nodes causes resource competition.	• Resource competition due to co-hosted applications • Input data skew • Remote input or output source being very slow • Faulty hardware

Internal reasons can be rectified by service provider, but external reasons cannot be resolved. The reasons are explained with an example, MapReduce cluster may be over-loaded with multiple tasks running on the same slave node. This would result in resource competition and would result in heterogeneous performance. The internal reasons can be resolved by allowing only one task to run on each slave node simultaneously or by allowing only different resource usage of tasks to share the same slave node.

D. Existing System

The previous speculative strategies begin only when a map task or reduce task is about to finish processing. Then it will select random number of tasks remaining and back up until all the available slots are allotted. This strategy does not consider problems like: i) whether the tasks running on straggler machine were really slow ? ii) Is the alternative slave node selected to run the backup task is efficient?

There are some drawbacks in the previous strategies. When the input data has some variably big files to process which cannot be distributed. The map tasks that process those records will process more data which will lead to data skew. Another problem is the phase completion percentage of the map and reduce tasks doesn't match. The reduce tasks starting to process before all the map tasks complete will also create problems. The reduce tasks will not complete unless outputs of all map tasks are available.

III. Problem Definition

The paper explains when a job is submitted to a MapReduce cluster, some of the nodes may be running very slow due to process overloading or hardware inefficiency. When multiple numbers of tasks are submitted in a cluster and almost all tasks completed but few tasks processing very slow resulting in overall delay of the cluster. This is the problem of this paper. The solution is to run those slow running tasks on other slave node in the cluster to get the better performance.

IV. Methodology

The new speculative execution strategy considers shortening of job execution time and improvement in the performance of the cluster. This strategy intends to select the slow running tasks accurately and back up them on proper worker nodes. This new speculation strategy chooses backup candidates based on the tasks process speed and remaining time of the tasks execution.

The speculation strategy uses EWMA (exponentially weighted moving average) method to predict the process speed. The EWMA scheme is explained in the next section.

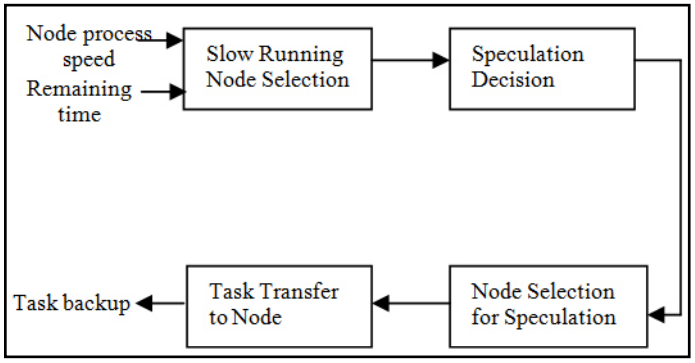

Fig 2 : Flow diagram of the speculation decision.

The major concern in this paper is reducing the job execution

time considering data locality that is we are making the data to be local for most of the tasks that are allocated to the cluster. Data locality is defined as how close compute and input data are, and it has different levels node-level, rack-level [9]. If a task is to be allocated a slot then it is checked to see if the slots are available in the local cluster, if it is available then the task will be allocated locally, if local slots are not available then it is checked to see if free slots are available in the group, and so on in the rack. If there are no slots locally available then the task is allotted anywhere in the cluster.

Data skew is another major concern in this paper to reduce the job execution time of the cluster. Tasks do not always process the same amount of data and may experience several types of data skew in MapReduce [6]. When the input data has few big input files that cannot be divided, the map tasks that process those records will have to process more data. When tasks are being assigned to map or reduce in the cluster the nodes map and reduce slots in the cluster are divide such a way that the high performing node will be assigned more jobs and the slow running node will be assigned the less of tasks to process. This method of variability in the task assignment can reduce the time required for speculation after a speculation task is executed.

A. Selection of slow running nodes

The speculation strategy selects the slow running (straggler machines) tasks based on the process speed and remaining time of the tasks. This paper uses EWMA algorithm to estimate the process speed.

B. EWMA(exponentially weighted moving average)

The EWMA is used to predict the process speed of the task. There are many prediction algorithms such as exponentially weighted moving average and CUSUM [7].

$E(t)= \alpha * O(t) + (1- \alpha) * E(t-1), \quad 0< \alpha<1, \quad (1)$

where E(t), O(t) is the estimated process speed and observed process speed at time t respectively. α is constant set to 0.2, where α reflects a tradeoff value between stability and responsiveness in the calculation. This is because when α is too low it cannot find the slow tasks and slow nodes in time while if it is too large, it may cause misjudgments.

C. Estimating the tasks remaining time and Backup time

In this paper the speculation strategy has the tendency to put the task with longest remaining time with highest priority to be backed up. A tasks remaining time is estimated by sum remaining time left in different phases in the map (reduce) task. In general, a map task is divided into map and combine phases, whereas reduce task is divided into copy, sort and reduce phases.

Reduce tasks can start when only some map tasks complete, which allows reduce tasks to copy map outputs earlier as they become available and hence mitigates network congestion. However, until all map tasks complete no reduce task can step into the sort phase. This is because each reduce task must finish copying outputs from all the map tasks to prepare the input for the sort phase.

The remaining time is calculated by adding remaining time of current phase and future phase in a map task.

$rem\_time = rem\_time_{cp} + rem\_time_{fp} \quad (2)$

To estimate the backup time for the slow task, we calculate sum of estimated time of each phase in this task as an estimation.

$backup\_time = \sum_p est\_time_p * factor_d \quad (3)$

D. Maximizing cost performance of cluster computing resources

Speculation execution has not only benefits, but also has resource utilization costs. In a Hadoop cluster the cost of speculation execution is task slots in the slave nodes, in this paper we are considering the benefit by time saved by speculation execution. Backing up a task will occupy two slots for backup_time (both the original and the backup need to keep running until either completes) and save one slot rem_time – backup_time. In contrast to without backing it up will cost just one slot of rem_time and benefit nothing. The profit of the two actions is:

$Profit_{backup} = \alpha*(rem\_time–backup\_time) –\beta*2*backup\_time. \quad (4)$

$profit_{not\_backup} = \alpha * 0 * - \beta*rem\_time. \quad (5)$

α, β are the weight of the benefit and cost, respectively.

Then the profit values of benefit of backup and benefit of not backup is compared. If the benefit of backing up a task is better than that of not backing up then that task is backed up otherwise it is left as it is to run in the same node.

V. Building Hadoop and Evelopment Environment

The paper is implemented in the CentOS operating system. To build the hadoop source it requires JDK environment. There are many packages that needs to be built before building the hadoop source they are: Cmake, OpenSSL, libtool, zlib, Protocol buffer, Apache Maven, Ant , Ivy. To build the hadoop source, we get hadoop build version in Github from the trunk and building hadoop jar files. Using maven we build the hadoop jar files.

VI. Results

The speculative execution strategy implemented in Hadoop will result in improvement in the performance of the Hadoop cluster. The Hadoop cluster keeps track of the status of each task in the cluster by heartbeat information. When a task is running very slowly in the cluster the speculation strategy will take decision to back up the slow running task in a node on to another node. The task in the slow node will be killed and that task is transferred to another node. The calculation of benefit of backup and not backup of tasks will consider time required to backup the task and remaining time to finish the task.

The speculation result of the task is shown in the task tracker. The tasks that are on the slow nodes are killed and backed up on another alternative node.

VII. Conclusion

In a Hadoop cluster of multiple slave nodes, some nodes may run considerably very slow causing the performance of the cluster throughput. In such cluster the slow running nodes are selected and a task will be backed up when it meets the following conditions: it has executed for a certain amount of time (i.e., the speculative lag), both the progress rate and the process bandwidth in the current phase of the task are sufficiently low the profit of doing the backup outweighs that of not doing it, its estimated remaining time is longer than the predicted time to finish on a backup node, it has the longest remaining time among all the tasks satisfying the conditions above.

VIII. Future Scope

The hadoop has great scope in distributed processing. The speculation strategy can be managed to be automated in assigning the tasks to better working nodes to improve the performance.

ISSN : 2394-2975 (Online)
ISSN : 2394-6814 (Print)

The slow working nodes can be given lower load to work so that it can be processed considerably better than node that backup the task.

IX. Acknowledgment

I am very thankful to my guide Mr Prashanth B. S. Assistant Professor, Department of Computer Science and Engineering, Mite for his cordial support, valuable information and guidance, to prepare this paper and also thankful to Prof. Dr. Nagesh H R, Head of the Department, Computer Science and Engineering, for his valuable and constructive suggestions during the planning and development of this work.

References

[1]. *J. Dean and S. Ghemawat, "Mapreduce: Simplified Data Processing on Large Clusters,"Comm. ACM, vol. 51, pp. 107-113, Jan. 2008.*
[2]. *Apache Hadoop, http://Hadoop.apache.org/ 2013.*
[3]. *M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: istributed Data-Parallel Programs from Sequential Building Blocks," Proc. Second ACM SIGOPS/EuroSysEuropean Conf. Computer Systems (EuroSys '07), 2007.*
[4]. *M. Zaharia, A. Konwinski, A.D. Joseph, R. Katz, and I. Stoica, "Improving Mapreduce Performance in Heterogeneous Environments," Proc. Eighth USENIX Conf. Operating Systems Design and Implementation, (OSDI '08), 2008.*
[5]. *G. Ananthanarayanan, S. Kandula, 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.*
[6]. *Y. Kwon, M. Balazinska, and B. Howe, "A Study of Skew in Mapreduce Applications," Proc. Fifth Open Cirrus Summit, 2011.*
[7]. *P.H. Ellaway, "Cumulative Sum Technique and Its Application to the Analysis of Peristimulus Time Histograms," Electroencephalography and Clinical Neurophysiology, vol. 45, no. 2, pp. 302-304, 1978.*
[8]. *A Study of Skew in MapReduce Applications YongChul Kwon, Magdalena Balazinska, Bill Howe University of Washington, USA, Jerome Rolia, HP Labs*
[9]. *Investigation of Data Locality in MapReduce Zhenhua Guo, Geoffrey Fox, Mo Zhou School of Informatics and Computing Indiana University Bloomington Bloomington, IN USA*

Authors Profile

Suprit S.H completed the bachelor's degree in Computer Science & Engineering from visvesvaraya technological University (VTU), currently pursuing Master degree in Computer Networks & Engineering at Mangalore Institute of Technology and Engineering under VTU Belgaum, Moodbidri.

Mr. Prashanth B.S completed bachelors and master degree in Computer Science and Engineering. Currently working as Assistant Professor in Mangalore Institute of Technology and Engineering, Moodbidri