ISSN 2395-1621

Hadoop's Configuration parameters Optimization Framework for Map Reduce Clusters

[#1]Miss. Trupti Mali, [#2]Prof. Deepti Varshney

[1]truptimali21@gmail.com.
[2]varshneydeepti11@gmail.com

[#12]Department of Computer Engineering
Shree Ramchandra College of Engineering Lonikand, Pune, India.

ABSTRACT

Most widely used frameworks for developing MapReduce-based applications is Apache Hadoop. But developers find number of challenges in the Hadoop framework , which causes problem to management of the resources in the Map Reduce cluster, that will optimize the performance of MapReduce applications running on it. Manually tuning configuration parameters is very time-consuming. Existing system uses Random forest approach, which automatically tune the Hadoop configuration parameters improving performance. Random forest approach try every combination of configuration parameter values and select the best one. This takes a considerable amount of time because of the huge number of Hadoop configuration parameter combinations. In the proposed system we consider the constraints in the resource allocation process in the MapReduce programming model for large-scale data processing for speed up performance. For that purpose we propose the novel technique called Dynamic approach for performing speed up of the available resources. It contains the two major operations; they are slot utilization optimization and utilization efficiency optimization. The Dynamic technique has the three slot allocation techniques they are Dynamic Hadoop Slot Allocation (DHSA), Speculative Execution Performance Balancing (SEPB), and Slot Pre-scheduling. It achieves a performance speedup by a factor of over the recently proposed cost-based optimization (CBO) approach. In addition performance benefit increases with input data set size.

Keywords: MapReduce, HDFS, Dynamic hadoop slot allocation, slot presheduling etc.

ARTICLE INFO

Article History

Received: 15th July 2017

Received in revised form :

15th July 2017

Accepted: 17th July 2017

Published online :

18th July 2017

I. INTRODUCTION

MapReduce is a popular computing framework for large-scale data processing. Practical experience shows that inappropriate configurations can result in poor performance of MapReduce jobs, however, it is challenging to pick out a suitable configuration in a short time. Also, current central resource scheduler may cause low resource utilization, and degrade the performance of the cluster. Appropriate configuration settings could reduce execution time of jobs by using cluster resources efficiently and avoiding unnecessary disk I/Os. Previous configuration tuning works can be categorized into three groups: following best practices and MapReduce tuning guides, offline configuration tuning, and online configuration tuning. Online tuning systems search appropriate configuration by dynamically assigning test configurations to running tasks in the job. However, there are multiple drawbacks in current online approach. Firstly, the searching strategies for finding optimal configuration take little consideration to characteristics of MapReduce; secondly, they neglect efficient resource utilization in the whole system; Thirdly, after a desirable configuration is achieved, the job uses the same configuration afterwards. However, the configuration might not be suitable for latter tasks because of data skew. Inappropriate configuration can cause a task being killed due to out of memory error. While tuning configuration parameters improves task performance, using cluster resources efficiently can also achieve significant performance improvement.

A. MapReduce Framework:

The mapreduce framework consists of two steps namely Map step and reduce step. Master node takes large problem input and slices it into smaller sub problems and distributes

these to worker nodes. Worker node may do this again and leads to a multi-level tree structure.

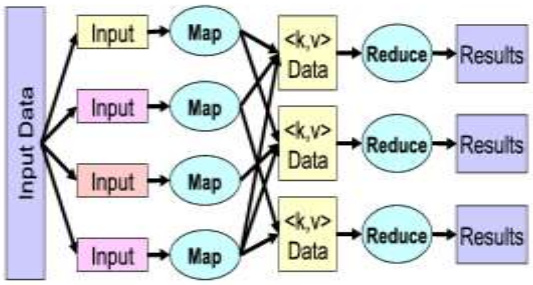

Fig.1 MapReduce Framework

Worker processes smaller problem and hands back to master. In Reduce step Master node takes the answers to the sub problems and combines them in a predefined way to get the output/answer to original problem. The MapReduce framework is fault-tolerant because each node in the cluster is expected to report back periodically with completed work and status updates.

If a node remains silent for longer than the expected interval, a master node makes note and re-assigns the work to other nodes. The detailed MapReduce framework is shown in Fig. 1

II. REVIEW OF LITERATURE

Zhendong Bei, Zhibin Yu, Huiling Zhang, Wen Xiong, Chengzhong Xu, Senior, "RFHOC is an automated performance tuning approach" that adjusts the Hadoop configuration parameters for an application running on a given cluster to achieve optimized performance. The model takes Hadoop configurations as input and outputs a performance prediction. In a subsequent step, we then use the performance prediction models for each phase as part of a genetic algorithm to search for the optimum Hadoop configuration for the application of interest.
This system automatically tune the Hadoop configuration parameters and build analytical models based on oversimplified assumptions, affecting the overall model's accuracy and ultimately the achievable performance improvements.[1]

Xiaoan Ding, Yi Liu, DepeiQian, "JellyFish:
Online Performance Tuning with Adaptive Configuration and Elastic Container in Hadoop Yarn", this paper proposes an online performance tuning system, JellyFish, to improve performance of MapReduce jobs and increase resource utilization in Hadoop YARN. JellyFish continually collects real-time statistics to optimize configuration and resource allocation dynamically during execution of a job. During performance tuning process, JellyFish firstly tunes configuration parameters by reducing the dimensionality of search space with a divideand- conquer approach and using a model-based hill climbing algorithm to improve tuning efficiency; secondly, JellyFish re-schedules resources in nodes by using a novel elastic container that can expand and shrink dynamically according to resource usage, and a resource re-scheduling strategy to make full use of cluster resources. The Proposed system does not support multiple resource rescheduling strategy and multi-tenant Hadoop environment. [4]

Dili Wu and AniruddhaGokhaleA ,"Self- Tuning System based on Application Profiling and Performance Analysis for Optimizing HadoopMapReduce Cluster Configuration" in this paper the PPABS framework comprises two distinct phases the Analyzer, which trains PPABS to form a set of equivalence classes of MapReduce applications for which the most appropriate Hadoop configuration parameters that maximally improve performance for that class are determined, and the Recognizer, which classifies an incoming unknown job to one of these equivalence classes so that its Hadoop configuration parameters can be self-tuned. Experimental results comparing the performance improvements for three different classes of applications running on Hadoop clusters deployed on Amazon Ee2 show promising results. Limitation of system despite its popularity, however, application developers face numerous challenges in using the Hadoop framework, which stem from them having to effectively manage the resources of a MapReduce cluster, and configuring the framework in a way that will optimize the performance and reliability of MapReduce applications running on it.[2]

Chi-Ou Chen, Ye-Qi Zhuo, Chao-Chun Yeh,
Che-Min Lin, Shih-wei Liao, "Machine Learning-Based Configuration Parameter Tuning on Hadoop System" In this paper, he focus on optimizing the Hadoop MapReduce job performance by tuning configuration parameters, and then we propose an analytical method to help system administrators choose approximately optimal configuration parameters depending on the characteristics of each application. approach has two key phases: prediction and optimization phase. The prediction phase is to estimate the performance of a MapReduce job, whereas the optimization phase is to search the approximately optimal configuration parameters strategically by invoking the predictor repeatedly. In our evaluation results, our work can help system administrators to improve the performance about 2X to 8X better than traditional methods. This current system is evaluated in cluster with the same nodes capability. [3]

Amelie Chi Zhou and Bingsheng, "HeTransformation-Based Monetary Cost Optimizations for Workflows in the Cloud", 2013., This paper proposes Transformation- based Optimization framework called as TOF for workflows in the cloud. TOF contains six basic workflow transformation operations. The transformation plan can be represented by the arbitrary performance and cost optimization process. In proposed TOF transformation based optimization framework is implemented. Its advantage is performance and cost optimization with the help of transformation sets and planner to guide the transformation It has two main process, they are transformation model and planner.
Transformation model means the set of transformation operations. Planner performs the transformation on the workflow based on the cost model.Recently, performance and monetary cost optimizations for workflows from various applications in the cloud have become a hot research topic. However, we find that most existing studies adopt ad hoc optimization strategies, which fail to capture the key

optimization opportunities for different workloads and cloud offerings (e.g., virtual machines with different prices). They consider only the single service provider. They used only the few transformation sets. [5]

Jorda Polo, Claris Castillo, David Carrera, Yolanda Becerra, Ian Whalley, MalgorzataSteinder, Jordi Torres, and Eduard Ayguad,"Resource-aware Adaptive Scheduling for MapReduce Clusters", This concept works for same workloads, but fails to capture the different resource requirements of individual jobs in multi-user environments. Their technique leverages job profiling information to dynamically adjust the number of slots on each machine, as well as workload placement across them, to maximize the resource utilization of the cluster. They present a resource-aware preparation technique for Map Re-duce multi-job workloads that aims at improving resource utilization across machines while observing completion time goals.

- They are not feasible due to number of the tasks.
- They may overload the system due to previous control cycle.
- They did not have enough memory for deploying the more memory tasks. [6]

Y Wang, "Budget-Driven Scheduling Algorithms for Batches of Map Reduce Jobs in heterogeneous Clouds", In this paper, they propose their task- level scheduling algorithms for Map Reduce workflows with the goals of optimizing budget and dead line constraints. They first consider the optimization problem under budget constraint where an in- stage local greedy algorithm is designed and combined with dynamic programming techniques to obtain an optimal global solution. To overcome the inherent complexity of the optimal solution, they also present two efficient greedy algorithms, called Global Greedy-Budget algorithm (GGB) and Gradual- Refinement algorithm (GR).In this paper, they studied two practical constraints on budget and dead
line of or the scheduling of a batch of Map Reduce jobs as a workflow on a se t of (virtual) machines in the Cloud .First, they focused on the scheduling-length optimization under budget constraints. They designed a global optimal algorithm by combining dynamic programming techniques with a local greedy algorithm for budget distribution on per stage basis, which was also shown to be optimal. It dynamically adjusts the size of a map task and assigns larger-size maps to the grid nodes with more powerful computing capabilities. Besides, it addresses the unevenly available bandwidth of a wide area network and avoids transferring large local regions owned by a single grid node to other nodes. [7]

M Hammoud, "Locality-Aware Reduce Task Scheduling for Map Reduce", 2011, Existing MapReduce schedulers define a static number of slots to represent the capacity of a cluster and create a fixed number of execution slots per machine. This abstraction works for homogeneous workloads, but fails to capture the different resource requirements of individual jobs in multi-user environments. Our technique leverages job profiling information to dynamically adjust the number of slots on each machine, as well as workload placement across them, to maximize the resource utilization of the cluster. In addition, our technique is guided by user-provided completion time goals for each job. Pioneer implementations of MapReduce have been designed to provide overall system goals. Thus, support for user-specified goals and resource utilization management have been left as secondary considerations at best. We believe that both capabilities are crucial for the further development and adoption of large-scale data processing. On one hand, more users wish for ad-hoc processing in order to perform short-term tasks. Therefore, providing consistency between price and the quality of service obtained is key to the business model of the Cloud. Resource management, on the other hand, is also important as Cloud providers are motivated by profit and hence require both high levels of automation and resource utilization while avoiding bottlenecks. In the proposed system, we present RAS, a Resource-aware Adaptive Scheduler for MapReduce capable of improving resource utilization and which is guided by completion time goals.
In addition, RAS addresses the system administration issue of configuring the number of slots for each machine and static solution for a multi-job MapReduce cluster. While the existing work focuses on the current typed-slot model wherein the number of tasks per worker is fixed throughout the lifetime of the cluster, and slots can host tasks from any job. It doesn't have the dynamic capacity control in their scheduler to adaptively change the allocations to meet higher level SLA goals such as deadlines. [8]

Ganesh Anantha narayanan, Srikanth Kandula, Albert Greenberg,"Reining in the Outliers in Map-Reduce Clusters using Mantri", 2010, Mantri identifies points at which tasks are unable to make progress at the normal rate and implements targeted solutions. If a task straggles due to contention for resources on the machine, restarting or duplicating it elsewhere can speed it up. If a task straggles due to contention for resources on the machine, restarting or duplicating it elsewhere can speed it up. The reason for poor performance is that they miss outliers that happen early in the phase and by not knowing the true causes of outliers, the duplicates they schedule are mostly not useful. [9]

III. SYSTEM ARCHITECTURE / SYSTEM OVERVIEW

Hadoop uses the slot-based resource model with the static configuration of map/reduce slots. There is a strict utility constrain that map tasks can only run on map slots and reduce tasks can only use reduce slots. Due to the rigid execution order between map and reduce tasks in a MapReduce environment, slots can be severely under-utilized, which significantly degrades the performance.

In contrast to YARN that gives up the slot-based resource model and propose a container-based model to maximize the resource utilization via unawareness of the types of map/reduce tasks, we keep the slot-based model and propose a dynamic slot utilization optimization system called DynamicMR to improve the performance of Hadoop by maximizing the slots utilization.
DynamicMR consists of three optimization techniques,

1) Dynamic Hadoop Slot Allocation (DHSA)

2) Speculative Execution Performance Balancing (SEPB)
3) Slot PreScheduling.

The performance and slot utilization of a Hadoop cluster can be optimized with the following step by step processes.

1) If a slot is idle, then DynamicMR will first attempt to improve the slot utilization with DHSA technique. It will evaluate based on numerous constraints like fairness, load balance and decide whether to allocate the idle slot to the task or not.
2) If the allocation is true, DynamicMR will further optimize the performance by improving the efficiency of slot utilization with SEPB. It works on top of Hadoop speculative scheduler to check whether to allocate the accessible idle slots to the pending tasks or to the speculative tasks.
3) When to allocate the idle slots for pending/ speculative map tasks, DynamicMR will be able to additional improve the slot utilization efficiency from the data locality optimization aspect with Slot Pre- Scheduling.

System architecture:

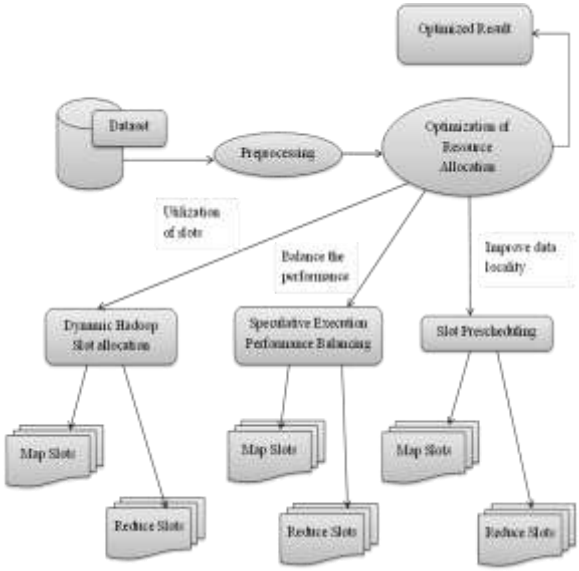

Fig 2. System architecture

Pre-processing Technique- Imputation:
Imputation is the process of replacing missing data with substituted values. Single imputation-A once common method of imputation was hot-deck imputation where a missing value was imputed from a randomly selected similar record. In cases with imputation, there is guaranteed to be no relationship between the imputed variable and any other measured variables. Thus, mean imputation has some attractive properties for univariate analysis but becomes problematic for multivariate analysis.

Slot allocation and Slot pre-scheduling process:
In this module we are going to perform two processes. Slot allocation Slot pre-scheduling process. In this slot allocation process we are going allocate the slot based on dynamic Hadoop slot allocation optimization mechanism. In the slot pre-scheduling process we are going to improve the data locality. Slot Pre-Scheduling technique that can improve the data locality while having no negative impact on the fairness of Map-Reduce jobs. Some idle slots which cannot be allocated due to the load balancing constrain during runtime, we can pre-allocate those slots of the node to jobs to maximize the data locality.

Hadoop Distributed File System (DFS):
Hadoop Distributed File System (DFS) will be configured for uploading the preprocessed geo data into hadoop. The configuration includes setting VM (hadoopplatform) IP and port for connection. FSDataInputStream and FSDataOutputStreamis used to FSDataInputStream and FSDataOutputStreamis used to upload and download data from hadoop. Different datacenters are analysed for data execution across different datacenters.

Speculative Execution Performance Balancing:
When a node has an idle map slot, we should choose pending map tasks first before looking for speculative map tasks for a batch of jobs. Hadoop Slot is executed for determining path for performing the MapReduce job. After this the Speculative based process starts to execute the determined optimized Multi execution path. Executing individual MapReduce jobs in each datacenter on corresponding inputs and then aggregating results is defined as MULTI execution path. This path used to execute the jobs effectively.

Performance Evaluation:
Speculative execution is a common approach for dealing with the straggler problem by simply backing up those slow running tasks on alternative machines. Multiple speculative execution strategies have been proposed, but there is a pitfall: incoming jobs are allocated to nodes present in server and fail to schedule process type allocate to node for processing. Performance is evaluated by means of selective the optimized resources and results taken in terms of execution time, processing memory etc.

IV. MATHEMATICAL MODEL

System Description:

S= {I, O, F, S1, S2}. where,

S = System.
I= Incoming job requests..
O= Job allocation with optimal resources.
F = {f1, f2, f3, f4}. where,

f1= Pre-processing.
f2= Slot allocation and slot pre-scheduling .
f3= HDFS Upload.
f4= Performance/ execution

S1= Initial state is the state in which system is waiting for incoming job requests.

S2= Final state is the Job allocation with optimal resources.

User Module:
Set U = u1, u2, u3, u4

u1=Available Hadoop Systems.
u2 = Resources Required
u3 =File for Execution.
u4 =analysis graph.

Set S = s1, s2, s3, s4
S1 = get user registration.
S2 = activate user for Hadoop services.
S3 = check the Files for Execution.
S4 = Allocate Resources for Execution.

Hadoop Module:
Set D = d1, d2, d3, d4, d5
d1 = Read file to be execute.
d2 = Analysis of File
d3 = Resource Allocation.
d4 = Execution.
d5 = Execution analysis.

V. SYSTEM ANALYSIS

After the analysis of the data, Hadoop is installed on a windows Machine and the sample data is inserted into the repository. And Later on, the Map-Reduce code is executed along with an initialization cluster. We can browse all the files in the default URL using the localhost. We need to specify all the paths for running the map-reduce like the input directory, output directory and the cluster initialization directory along with the algorithm we used. Hadoop provides a reliable output in mapping and analyzing of the data. This Data can be employed for further representation.

System requirement:

We have created system in java. Data is processing on hadoop environment windows system. We have created a java application with local server.java application that communicates with local server and Trustee Server using REST API. We have uploaded text document on hadoop. We have evaluated time, speedup, automatically processing required for uploaded dataset. Here we also calculate the load each datacenter shown fig3. analysis purpose.

Algorithm:

Input: A set J of n MapReduce jobs. Di is the attribute of job Ji as defined above.

Output: Schedule σ (order of jobs execution.)

1: Sort the original set J of jobs into the ordered list L using their stage duration attribute D^1i
2: head ← 1,tail ← n
3: for each job Ji in L do
4: if D^2i = m then
5: // Put job Ji from the front
6: σ head ← Ji, head ← head + 1
7: else
8: // Put job Ji from the end
9: σ tail ← Ji, tail ← tail - 1
10: end if
11: end for

Fig 3. Each data center load will analysis

After execution of the jobs we calculate the final analysis result of the how much load can perform the each data center for further analysis. We analysis the above graph and calculate load on DC.

VI. RESULT TABLE

Parameter	Before Processing	After Processing
Size of the File	183432 BYTES	35765 BYTES
No. of Records	409	76
No. of Attribute	77	75

Table I. Result analysis before and after execution on hadoop environment

VII. CONCLUSION

This paper proposes a DynamicMR Technique can be use to improve the execution of MapReduce workloads while charge up the fairness. It comprises of three methods, in exacting DHSA, SEPB and Slot PreScheduling, all of which consider on the slot use optimization for MapReduce assembly from alternate extension by distributing map or reduce slots to map and reduce tasks alterably.

ACKNOWLEDGMENT

I wish to express my profound thanks to all who helped us directly or indirectly in making this paper. Finally I wish to thank to all our friends and well-wishers who supported us in completing this paper successfully I am especially grateful to our guide Prof. Deepti Varshney Sir for him time to time, very much needed, valuable guidance. Without the full support and cheerful encouragement of my guide, the paper would not have been completed on time.

REFERENCES

[1] Zhendong Bei, Zhibin Yu, Member, IEEE, Huiling Zhang, Wen Xiong,"RFHOC: A Random-Forest Approach to Auto-Tuning Hadoop's Configuration", JOURNALOF L

ATEX CLASS FILES, VOL. 6, NO. 1, IEEE, JANUARY 2007.

[2] Dili Wu and AniruddhaGokhaleA ,"Self-Tuning System based on Application Profiling and Performance Analysis for Optimizing HadoopMapReduce Cluster Configuration" ISIS, Dept of EEeS, Vanderbilt University, 1025 16th Ave S, Nashville, TN 37212, USA, IEEE,2013.

[3] Chi-Ou Chen, Ye-Qi Zhuo, Chao-Chun Yeh, Che-Min Lin, Shih-wei Liao, "Machine Learning-Based Configuration Parameter Tuning on Hadoop System", 978-1-4673-7278-7/15, IEEE,2015.

[4] Xiaoan Ding, Yi Liu, DepeiQian, "JellyFish: Online Performance Tuning with Adaptive Configuration and Elastic Container in Hadoop Yarn", 1521-9097/15, 2015 IEEE 2015.

[5] Coupling task progress for Map Reduce resource aware scheduling. J. Tan, X. Q. Meng, L. Zhang.

[6] Map-Reduce: Simplified Data Processing on Large Clusters J. Dean and S. Ghemawat.

[7] Reining in the Outliers in Map-Reduce Clusters using Mantri, Ganesh Ananthanarayanan,SrikanthKandula, Albert Greenberg.

[8] Resource-aware Adaptive Scheduling for Map-Reduce Clusters. J. Polo, C. Castillo, D. Carrera.

[9] Two Sides of a Coin: Optimizing the Schedule of MapReduce Jobs to Minimize Their Makespan and Improve Cluster.

