ISSN 2395-1621

# Optimization Framework for Map Reduce Clusters on Hadoop's Configuration

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

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

[#1]Department of Computer Engineering
[#2]HOD, Department of Computer Engineering

Shree Ramchandra College of Engineering, Lonikand,
Pune, India

**ABSTRACT**

**Hadoop represents a Java-based distributed computing framework that is designed to support applications that are implemented via the MapReduce programming model. Hadoop performance however is significantly affected by the settings of the Hadoop configuration parameters. Unfortunately, manually tuning these parameters is very time-consuming. Existing system uses Random forest approach, which automatically tune the Hadoop configuration parameters for optimized performance for a given application running on a given cluster. Random forest approach try every combination of configuration parameter values and choose the best one. Unfortunately, this is unrealistic because of the huge number of Hadoop configuration parameter combinations. This takes a considerable amount of time, leading to impractically long times. 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 we proposed 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 Prescheduling. 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: Map Reduce, Hadoop, Clustering, HDFS.**

ARTICLE INFO

**Article History**

Received: 4th December 2016

Received in revised form :

4th December 2016

Accepted: 8th December 2016

**Published online :**

**8th December 2016**

## I. INTRODUCTION

MapReduce provides an effective solution to parallel program development and massive data processing on distributed platforms. As an open source implementation, Hadoop is widely used in building MapReduce-based applications on large clusters. Despite the popularity and usability of Hadoop, application developers and system users face a series of challenges to achieve good performance in their applications. It often requires specialized system knowledge and tuning skills to obtain appropriate configuration [2]. Researchers have shown that Hadoop configuration plays an important role in performance of MapReduce programs. Appropriate configuration settings could reduce execution time of jobs by using cluster resources efficiently and avoiding unnecessary disk I/Os. Moreover, some parameters decide if a job can be successfully executed and should be treated carefully. However, it is difficult to obtain an optimized configuration because: (1) there exists hundreds of parameters in the system; (2) parameters are related each other and act cooperatively; (3) configuration is application and hardware dependent, that is, optimized configuration is specific to characteristics and input dataset of an application for specified cluster.

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. Researchers have shown that average resource utilization in real-world data centers is fairly low. Generally reasons for low resource utilization includes: (1) tasks request more resources than they actually need; (2) resource usage varies during task execution but the amount of resources allocated to a container is fixed; (3) rest resources in a node is not enough for new containers and remain idle.

## II. LITERATURE SURVEY

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.

**Limitation:** 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]

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 called the Analyzer, which trains PPABS to form a set of equivalence classes of MapReduce applications for which the most appropriate Hadoop config-uration 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:** 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 HadoopMapReduce 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 evaluationresults, our work can help system administrators to improve the performance about 2X to 8X better than traditional methods.

**Limitation:** The prediction phase is to estimate the performance of a MapReduce job.[3]

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 divide-and-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.

**Limitation:** Experimental results show that JellyFish can improve performance of MapReduce jobs by an average of 24% for jobs run for the first time, and by an average of 65% for jobs run multiple times compared to default YARN. [4]

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).

**Limitation:** 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.

**Limitation:**

- 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.

**Limitation:** 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.

**Limitation:** 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.

**Limitation:** 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]

Jeffrey Dean and Sanjay Ghemawat , "Map-Reduce: Simplified Data Processing on Large Clusters", 2004,MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key. Many real world tasks are expressible in this model, as shownin the paper. Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The run-time system takes care of the details of partitioning the input data, scheduling the program's execution across a set of machines, handling machine failures, and managing the required inter-machine communication. This allows programmers without any experience with parallel and distributed systems to easily utilize the resources of a large distributed system. Our implementation of MapReduce runs on a large cluster of commodity machines and is highly scalable: a typical MapReduce computation processes many terabytes of data on thousands of machines. Programmers _nd the system easy to use: hundreds of MapReduce programs have been implemented and upwards of one thousand MapReduce jobs are executed on Google's clusters every day.

**Limitation:** The cluster consists of hundred of thousands of machines, and therefore machine failures are common.[10]

## III. BASIC CONCEPT

**Map Reduce:**

Map Reduce is a processing technique and a program model for distributed computing based on java. It contains two important tasks, namely Map and Reduce. The major advantages of MapReduce is that it is easy to scale data processing over multiple computing nodes [4].

**Hadoop Distributed File System(HDFS):**

It is distributed file system designed to run on commodity hardware. This system provides high-throughput access to application data. HDFS is highly fault- tolerant and is designed to be deployed on low-cost hardware. Application that run on HDFS has large data sets. Typically file in HDFS is gigabytes to terabytes in size [5]. It should support tens of millions of files in a single instance. HDFS is designed more for batch process in gather than interactive use by users. Detection of faults and quick, automatic recovery from them is a core goal of HDFS. HDFS has been designed to easily portable from one platform to another. HDFS has a Master-slave architecture. An HDFS cluster consist of a single Name Node, a master serves that manages the file system namespaces and regulates access to files by clients. In addition, there are number of data nodes, usually one per node in the cluster, which manage storage attached to the nodes that they run on [4].

## IV. EXISTING SYSTEM

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.

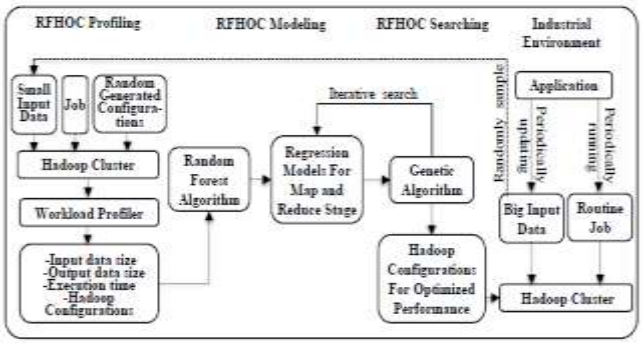


Fig 1. Existing System

**Problem Statement:**

Existing 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 [2]. Performance tuning is a challenging problem for Hadoop/MapReduce workloads because of the large number of Hadoop configuration parameters. Previously proposed techniques 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.

## V. PROPOSED SYSTEM

We proposed alternate technique to improve performance speedup using three techniques. Dynamic slot allocation Scheme (DSAS), it allows slot to reallocate or reduce slots based on its need. Execution Performance balancing Scheme to balance the performance criteria between single job and group of jobs. Slot Pre-scheduling technique that can perform data locality with cost fairness. Thus we produce a system called Dynamic MR to improve the performance Map Reduce data set significantly.

In the dynamic process apart from the three concepts present in the paper we are going to introduce clustering approach. In addition to the multi data center processing we are going to add clustering concept. Because we are going to split the data and process the data in multiple datacenters. If we combine the similar data's into clusters algorithm. By clustering the data we

 

can able to process the data in short execution time. After preprocessing, we split our process into multiple files and apply clustering process. Here we are going to use clustering algorithm which is a standard algorithm, which helps to process the data in a short execution time and also improve the performance speed up of the system.

System architecture:

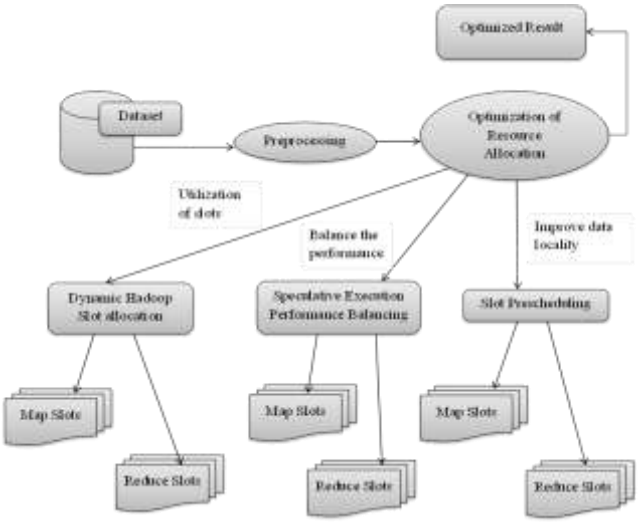


Fig 2. System architecture

## VI. METHODOLOGY

**Preprocessing 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 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.

## VII. FUTURE SCOPE

In future work we will work on hadoop environment to improve resource allocation and performance speed up. And we also check the performance of hadoop on cloud environment.

## VIII. CONCLUSION

In this project we study smart data processing using the resource allocation process in the MapReduce programming model for large-scale data processing. We perform data center resizing and data routing to reduce the operational cost in datacenters for big data processing. And also minimize the cost of data center and improve the performance and speedup.

## REFERENCE

[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.

[10] B. Moseley, A. Dasgupta, R. Kumar, T. Sarl, On scheduling in map-reduce and flow-shops. In SPAA'11, pp. 289-298, 2011.

[11] C. O˘guz, M.F. Ercan, Scheduling multiprocessor tasks in a two-stage flow-shop environment. Proceedings of the 21st international conference on Computers and industrial engineering, pp. 269-272, 1997.

[12] B. Palanisamy, A. Singh, L. Liu and B. Jain, Purlieus: Localityaware Resource Allocation for MapReduce in a Cloud, In SC'11, pp. 1-11, 2011.

[13] J. Polo, C. Castillo, D. Carrera, et al. Resource-aware Adaptive Scheduling for MapReduce Clusters.In Middleware'11, pp. 187-207, 2011.

[14] J. Tan, X. Q. Meng, L. Zhang. Coupling task progress for Map Reduce resource aware scheduling.In IEEE Infocom'13, pp. 1618-1626, 2013.

[15] J. Tan, S. C. Meng, X. Q. Meng, L. Zhang. Improving ReduceTask data locality for sequential MapReduce jobs.In IEEE Infocom'13, pp. 1627-1635, 2013.

 