ISSN 2395-1621

Efficient framework for optimized allocation of slots for map reduce mechanism

[#1]Dhananjay Ahire, [#2]Narayan Dongare, [#3]Pooja Wankhede, [#4]Chiranjivi Mane

[1]ahire.dhananjay12@gmail.com,
[2]narayandongare7784@gmail.com,
[3]chiranjivimane @gmail.com,
[4]poojaw2903@gmail.com

[#1234]Deparment of Computer Engineering
JSPM's, ICOER, Wagholi, Pune.

ABSTRACT

MapReduce is a current computing standard for important large data processing in one or more clusters. A MapReduce system having a set of jobs and each job having a multiple map and multiple reduce tasks because of these 1) in map slot, map task are get run and in reduce slot, reduce task are get run, and 2) reduce tasks are executed after map tasks, execution orders of job is different and map/ reduce slot configurations for different system have a different execution performance and utilization. But those existing techniques in the process are not providing the better performance because of the un-improved resource allocation. To overcome this drawback, and to improve a performance of resource allocation with the proposed technique called DynamicMR. this concept includes a two phases of algorithm is MK_TCT (makespan and the total completion time) for an offline workload and second phase of algorithm for optimized resource allocation . In these system to consider a three major steps to improve a system performance and resource utilization for basic MapReduce technique, 1) DHSA- Dynamic Hadoop Slot Allocation (i.e it used to overcome the slot allocation restraint), 2)SEPB- Speculative Execution Performance Balancing (i.e it used to steadiness the performance in cluster of jobs), 3)SP-Slot Pre-scheduling (i.e it used to advance the data vicinity). Micro-level optimization is used as a optimization approach, it is used to improve idle slot utilization efficiency after the Macro-level optimization. Particularly, we identify that three main affecting factors: Speculative tasks, dynamic slot allocation, and pre-scheduling.

Keywords: MapReduce, Dynamic Hadoop Slot Allocation [DHSA], Job ordering, Scheduling algorithm, Speculative , Execution Performance Balancing [SEPB], Slot Pre-scheduling, MK_TCT, DynamicMR , Optimized Resource Allocation, Flow-shops.

ARTICLE INFO

Article History

Received: 15th May 2017

Received in revised form :

15th May 2017

Accepted: 22nd May 2017

Published online :

23rd May 2017

I. INTRODUCTION

Over the past five years, the authors and many others at Google have executed hundreds of special-purpose multiplications that process large amounts of raw data, such as crawled papers, web request logs, etc., to compute various kinds of derived data. However, the input data is usually large and the deductions have to be distributed across hundreds or thousands of machines in order to surface in areas on able quantity of time. The subjects of how to parallelize the calculation, allocate the data, and handle failures conspire to obscure the original simple calculation with large quantities of multifaceted code to deal with these issues. Firstly, the compute resources (e.g., CPU cores) are abstracted into map and reduce slots, which are basic compute units and statically configured by administrator in advance. A MapReduce job execution has two unique features:

1) The slot allocation constraint assumption that map slots can only be allocated to map tasks and reduce slots can only be allocated to reduce tasks, and

2) The general execution constraint that map tasks are executed before reduce tasks.

Due to these features, we have two observations: (I). there are significantly different performance and system utilization for a MapReduce workload under different slot

configurations, and (II). even under the optimal map/reduce slot configuration, there can be many idle reduce (or map) slots while map (or reduce) slots are not enough during the computation, which adversely affects the system utilization and performance. Speculative execution is an important technique that can overcome the problem of slow-running task's influence on a single job's execution time by running a backup task on another machine. However, it comes at the cost of cluster efficiency for the whole jobs due to its resource competition with other running tasks. We propose a dynamic scheduling and slot allocation policy for speculative task. It balances the tradeoff between a single job's execution time and a batch of jobs' execution time by determining dynamically when it is time to schedule and allocate slots for speculative tasks.

sDelay scheduling has been shown to be an effective approach for the data locality improvement in MapReduce. It achieves better data locality by delaying slot assignments in jobs where there are no currently local tasks available. However, it is at the cost of fairness. In contrast, we propose a scheduler named Pre-Scheduler that can improve the data locality while not hurt the fairness. It considers the case of a node where there are currently local map tasks of jobs and idle slots available, but no allowable idle map slots (e.g., due to the load balancing constrain) to be allocated. It pre-allocates idle slots of the node to jobs to maximize the data locality and guarantee fairness. Since Delay scheduling and Pre-scheduling works in different scenarios, our DLMS incorporates both approaches, making them work cooperatively to maximize the data locality.

We propose DynamicMR, a dynamic scheduling framework for MapReduce, in order to improve the utilization and performance of a shared Hadoop cluster under a fair scheduling between users. Figure 1 gives an overview of DynamicMR. It consists of three levels of scheduling components, i.e., Dynamic Hadoop Fair Scheduler (DHFS) , Dynamic Speculative Task Scheduler (DSTS) , and Data Locality Maximization Scheduler (DLMS) .Each scheduler considers the performance improvement from different aspects. DHFS attempts to maximize slot Utilization as possible while guarantee the fairness, when there are pending tasks (e.g., map tasks or reduce tasks).DSTS identifies the slot resource in-efficiency problem for a Hadoop cluster, brought by speculative tasks.

1.1 Job Ordering Optimization

We can implement the job ordering algorithm like MK_JR to improve the efficiency and performance of a system. Implement the MK_JR using johnson's rule[4][5]. For MK_TCT implement MK_JR. It turns out to be equivalent to the two stage flow shop problem when there one map slot and one reduced slot only.

Johnson's Rule [4]can produced optimal result for job order makespan in this case. Maximizing the makespan in this case is NP-Hard.

1.2 Slot Configuration Optimization

It plays important role in to improve performance and efficiency and it have significant impact on performance for MapReduded Workload. Significant impact shows is in (Fig.2a), performance difference in worst case and optimized configuration.

Here we purposed a enumeration algorithm for MapReduced Slot configuration and It gives good result performance than previous state.

Experimental Result: Result are come more better than previous System. We evaluate the result through a Simulation of Workload on single Datanode and we observe and experiments show that,

1. For Makespan, the job ordering optimization algorithm archives an approximately10 -15% improvement.
2. For Makespan, The slot configuration optimization algorithm archive 4-5X great result.
3. For complete total time [TCT]:
   5X improvement with bi-directional MapReduced Job Ordering Optimization
   And 4X improvement with bi-directional MapReduced Slot Config. Optimization

II. MOTIVATION

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.

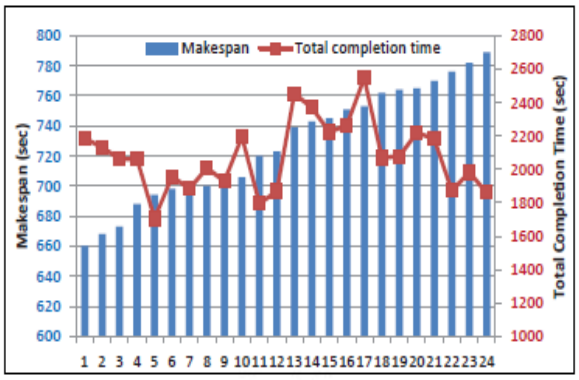

Fig 2a.job orders Id result of MapReduced Workload

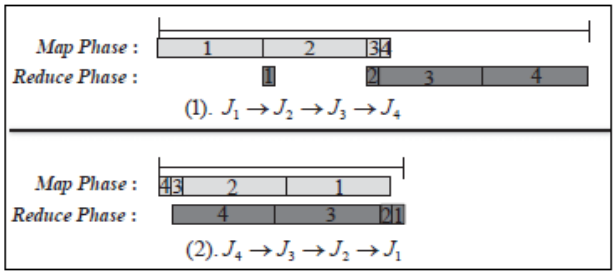

Fig 2.MapReduce execution flow for different job orders

If we want to improve the performance of the previous system then we need to focus on these two points like, 1) how to manage a slots or configure a slots and 2) how to prescheduled or how to manage the order of jobs.

So to get the good solution performance so we need to implement the **1) Job Ordering Optimization & 2) Slot Configuration Optimization.** After implementing these thing we defiantly gets good results.

In Above figure 2a and 2b we observe one thing, to get good performance we need to focus on these two basic things.

III. SYSTEM OVERVIEW

A. Data preprocessing

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.

B. Dynamic Hadoop Slot Allocation(DHSA)

In contrast to YARN which proposes a new resource model of container that both map and reduce tasks can run on, DHSA keeps the slot-based resource model. The idea for DHSA is to break the assumption of slot allocation constraint to allow that: 1) Slots are generic and can be used by either map or reduce tasks, although there is a pre-configuration for the number of map and reduce slots. In other words, when there are insufficient map slots, the map tasks will use up all the map slots and then borrow unused reduce slots Similarly, reduce tasks can use unallocated map slots if the number of reduce tasks is greater than the number of reduce slots. 2)Map tasks will prefer to use map slots and likewise reduce tasks prefer to use reduce slots. The benefit is that, the pre-configuration of map and reduce slots per slave node can still work to control the ratio of running map and reduce tasks during runtime, better than YARN which has no control mechanism for the ratio of running map and reduce tasks. The reason is that, without control, it easily occurs that there are too many reduce tasks running for data shuffling, causing the network to be a bottleneck seriously

C. Speculative Execution Performance Balancing (SEPB)

When a node has an idle map slot, we should choose pending map tasks first before looking for speculative maptasks 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.

D. Slot pre-scheduling(SP)

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.

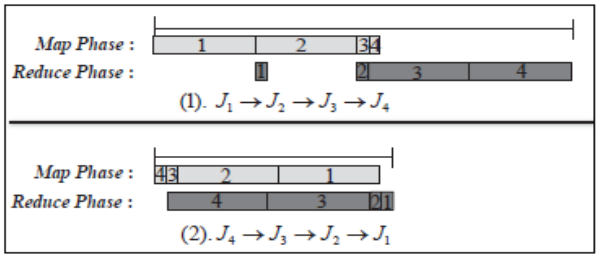

Fig.3: Slot Scheduling for map and reduce task

E. 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.

IV. USEFUL ALGORITHMS

Following algorithm plays important role in the Dynamic MapReduced.

A. Make Span Optimization
B. MK_TCT
C. MK_JR
D. FIFO
E. Job Ordering Optimization

V. SOFTWARE REQUIREMENT SPECIFICATION

Following tools and setups are required for the project:

- **VMware Player:**
  To run hadoop appliance file and to create virtual server for windows machine

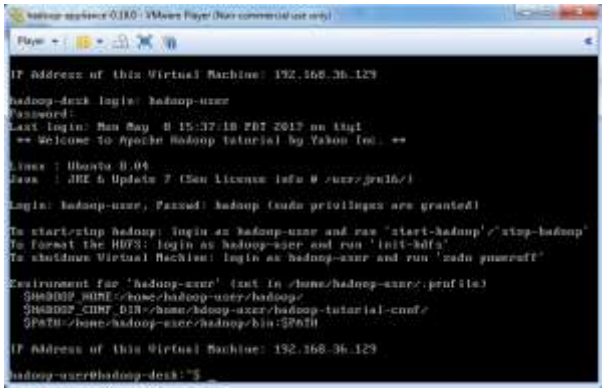

Fig.3: VM player through appliance file loaded

- **Cygwin Terminal:**
  To get a linux flavor in windows. We can use a cmd prompt as terminal because of cugwin terminal.

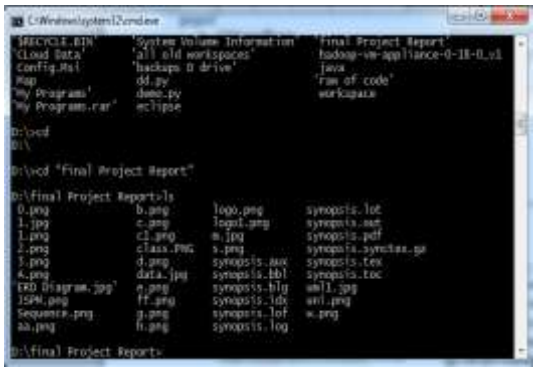

Fig.3: cmd prompt work like terminal

- **Java6 (J2SE And J2EE)**

  Java Standard and Enterprise Edition is used to code for project. Each and everything is coded neatly and easily through J2SE and J2EE.

- **Eclipse3.3.1 or 3.3.3.1**

  These is a eclipse set up which is used for the java 1.6 version or before it. Its easy to run a map reduced program and build a map reduced program.

- **Hadoop installation and Server creation:**

  We need to create hadoop server through the eclipse DFS server creation. We easily can create a server using dfs location.

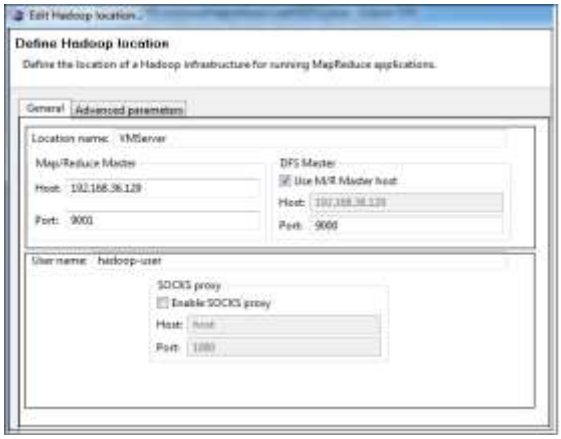

Fig.3: Hadoop DFS location creation

VI. MATHEMATICAL MODEL

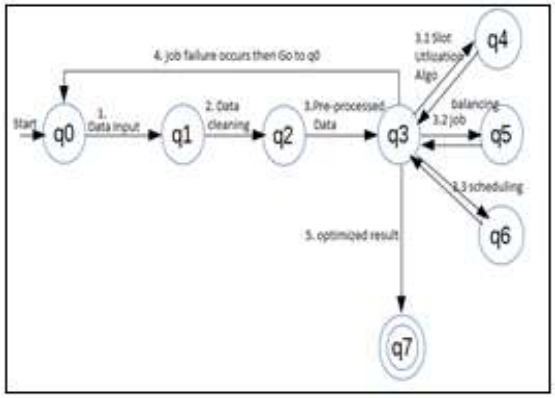

Fig.3: Mathmatical model of Dynamic MR

Where,
q0- Initial State
q1- Data cleaning
q2- Data preprocessing
q3- Optimization of resource allocation
q4- Slot utilization process
q5- Speculative execution of jobs
q6- Job pre-scheduling
q7- Final optimized result

VII. SYSTEM ANALYSIS

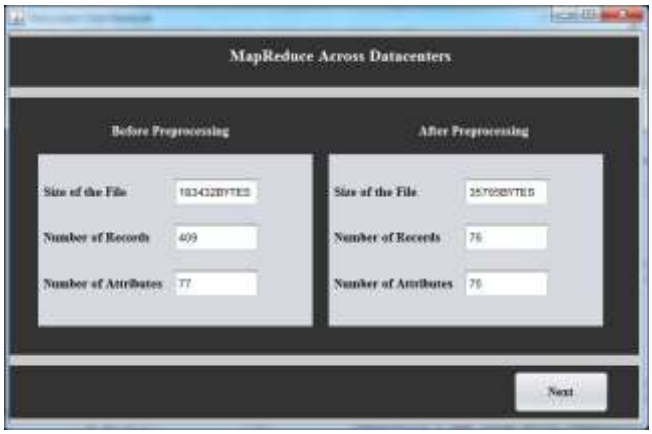

Fig.3: file sized before and after preeprocessing

Fig.3: Slot allocation for maper and reducer

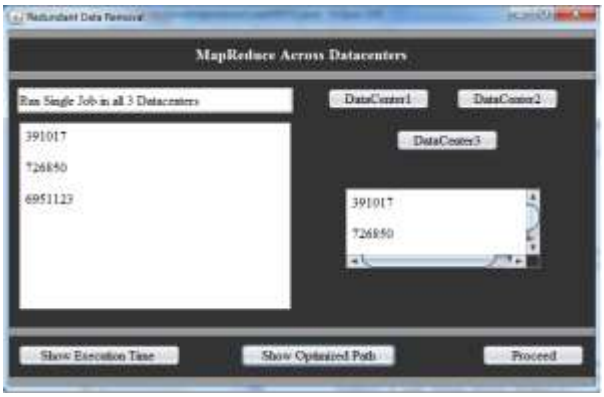

Fig.3: Execution of map reduce on different data center

Fig.3:Optimized result set

VIII. CONCLUSION

The aim of the proposed system is to improve the performance of Map Reduce workloads. It considered three techniques: Dynamic Hadoop Slot Allocation Speculative Execution Performance Balancing, and Slot Pre-scheduling. Dynamic Hadoop Slot Allocation use allocation of map to maximize the slot utilization and it reduces the task dynamically.

IX. ACKNOWLEDGMENT

With Deep Sense of gratitude we would like to thank all the people who have lit our path with their kind guidance. We are very thankful and grateful to these intellectuals who did their best to help during our project work.

We would like to take this opportunity to thank our internal guide **Prof. D. P. Gadekar** for giving us all the help and guidance we needed. We are really grateful to them for their kind support. Their valuable suggestions and were being very helpful. Without the full support and cheerful encouragement of my guide, the paper would not have been completed on time.

REFERENCES

[1] Simplied Data Processing on Large Clusters, In Proceedings of the 6th Symposiumon Operating SystemsDesign and Im-plementation (OSDI), J. Dean and S. Ghemawat, 2004.
[2] Hadoop. http://hadoop.apache.org.
[3] On scheduling in map-reduce and ow-shops. SPAA, pp. 289-298, B. Moseley, A. Dasgupta, R. Kumar, T. Sarl, 2011.
[4] Two Sides of a Coin: Optimizing the Schedule of MapReduce Jobs to Minimize Their Makespan and Improve Cluster Per-formance. MASCOTS, A. Verma, L. Cherkasova, R. Campbell,2012.
[5]Making a Yellow Elephant Run Like a Cheetah, PVLDB J. Dittrich, A. Quiane-Ruiz, A. Jindal, Y. Kargin, V. Setty, and J. Schad. Hadoop, 2010.
[6] The Performance of MapReduce: An Indepth Study, PVLDB, 3:472-483D.W. Jiang, B.C. Ooi, L. Shi, and S. Wu, 2010.
[7] MRShare: Sharing Across Multiple Queries in MapReduce Proc. of the 36th VLDB (PVLDB), Singapore, September T. Nykiel, M. Potamias, C. Mishra, G. Kollios, and N. Koudas. 2010.
[8]http://wiki.apache.org/hadoop/HowManyMapsAndReduces.
[9]Scheduling for Multi-user Mapreduce Clusters. M. Zaharia, D. Borthakur, J. Sarma, K. Elmeleegy,S. Schenker,I. Stoica, Job Technical Report EECS-2009-55, UCBerkeley Technical Report (2009).
[10]http://hadoop.apache.org/common/docs/r0.20.1/capacity/scheduler.html Capacity Scheduler Guide. 2010.
[11] J.N.D. Gupta, Two stage hybrid owshop scheduling problem, Journal of Operational Research Society 39 (4) (1988) 359C364.
[12]When the herd is smart: aggregate behavior in the selection of job request,IEEE Transactions on Parallel and Distributed Systems, vol. 14, pp.181-192, 2003, W. Cirne and F. Berma.
[13]Optimal two- and three-stage production schedules with setup times included. Naval Res Logist Q. vol. 1, pp. 61-68, 1954, S.M. Johnson.
[14] Scheduling Shared Scans of Large Data Files. In VLDB, 2008, P. Agrawal, D. Kifer, and C. Olston.
[15]http://en.wikipedia.org/wiki/Lognormal distribution.
[16] Amazon EC2. http://aws.amazon.com/ec2.
[17]Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling In Proceedings of EuroSys, M. Zaharia, D. Borthakur, J. Sarma, K. Elmeleegy,S. Schenker,I. Stoica-2013.
[18] http://hadoop.apache..org to understand a basic things for projects.

[19]http://riot.ieor.berkeley.edu/Applications/Schedu ling/ algorithms.html,The Scheduling Problem.
[20]A model of computation for MapReduce, Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 938-948,K. Howard, S. Siddharth and V. Sergei, 2010

