Minimizing the MakeSpan of Multiple MapReduce Jobs through Job Ordering Technique

[1]EDUKOJU HAREESH, [2]N. NAVEEN KUMAR

[1]M.Tech Student, Department of CSE, JNTU, Kukatpally, Hyderabad.

[2]Assistant Professor, Department of CSE, JNTU, Kukatpally, Hyderabad.

Abstract— ***In today's world the amount of data being generated is growing exponentially. A number of these data are structured, semi-structured or unstructured. This poses an excellent challenge once these information are to be analyzed as a result of conventional data processing techniques are not suited to handling such information. Map reduce may be a programming model and an associated implementation for process and generating massive information sets. A Map reduce workload usually contains a group of jobs, every of that consists of multiple map tasks followed by multiple reduce tasks. This technique proposes of algorithms to optimize the Makespan and also the total completion time for an offline MapReduce workload. Our algorithms concentrate on the task ordering optimization for a MapReduce workload and that we will perform optimization of Makespan and total completion for a MapReduce workload. Our work is focuses on resolution the time efficiency issues still as memory utilization problem. By using MK_TCT_JR algorithm made the result that are up to, 90 to fix things than MK_JR. Our algorithm can improve the system performance in terms of Makepan and total completion time.***

Index Terms -- ***Mapreduce, Hadoop, Scheduling Algorithm, Job Ordering, Makespan, Total Completion Time, Resource Allocation Mapreduce Slot Allocation;***

I. INTRODUCTION

A MapReduce job consists of a group of map and reduce tasks, wherever reduce tasks are performed when the map tasks. Hadoop, an open source implementation of MapReduce, has been deployed in giant clusters containing thousands of machines by companies like Amazon and Facebook. Make span and total completion time are two key performance metrics. Generally, build span is defined because the period of time since the beginning of the primary job until the completion of the last job for a group of jobs. It considers the computation time of jobs and is usually used to measure the performance and utilization efficiency of a system. In distinction, total completion time is referred to because the ad of completed time periods for all jobs since the beginning of the primary job. It's a generalized build span with queuing time (i.e., waiting time) included. We are able to use into measure the satisfaction to the system from one job's perspective through dividing the entire completion time by the quantity of jobs (i.e., average completion time). Therefore, during this paper, we tend to aim to optimize these two metrics Objectives: to enhance the performance for MapReduce workloads with job ordering and slot configuration optimization approaches. Recommend slot configuration algorithms for total completion time and make span. Execute exhaustive experiments to authenticate the effectiveness of projected algorithms and theoretical results. In recent

years, MapReduce has become the parallel computing paradigm of alternative for large-scale data processing in clusters and data centers. A MapReduce job consists of a group of map and reduce tasks, wherever reduce tasks are performed when the map tasks. Hadoop, an open source implementation of MapReduce, has been deployed in giant clusters containing thousands of machines by companies such as Yahoo! and Facebook to support execution for big jobs submitted from multiple users (i.e MapReduce workloads). During a Hadoop cluster, the calculate resources are abstracted into map (or reduce) slots, that are basic calculate units and statically organized by administrator earlier. attributable to 1) the slot allocation constraint assumption that map slots will only be allotted to map tasks and reduce slots will only be allotted to reduce tasks, and 2) the overall execution constraints that map tasks are executed before reduce tasks, we have two observations: (I). there are significantly totally different performance and system utilization for a MapReduce workload below different job execution orders and map/reduce slots configurations, and (II). Even below the optimal job submission order as well because the optimal map/reduce slots configuration, there may be several idle reduce (or map) slots whereas map (or reduce) slots are not enough throughout the computation, that adversely affects the system utilization and performance. In our work, we tend to address the matter of a way to improve the employment and performance of MapReduce cluster without any previous data or data (e.g., the coming time of MapReduce jobs, the execution time for map or reduce tasks) concerning MapReduce jobs. Our resolution is novel and straightforward: we tend to break the previous initial assumption of slot allocation constraint to allow one. Slots are generic and might be utilized by map and reduce tasks. 2. Map tasks can prefer to use map slots and likewise reduce tasks prefer to use reduce slots. In alternative words, once there are insufficient map slots, the map tasks can assign all the map slots and so borrow unused reduce slots. Similarly, reduce tasks will use unallocated map slots if the quantity of reduce tasks is greater than the quantity of reduce slots. During this paper, we are going to focus specifically on Hadoop fair scheduler (HFS). This can be as a result of the group exploitation and concert for the complete MapReduce jobs below HFS are abundant poorer (or a lot of serious) than that below FIFO scheduler. However it is value mentioning that our resolution may be used for FIFO scheduler as well.

II. RELATED WORK

In literature, there was research study on performance optimization of Hadoop MapReduce jobs. An essential approach for upgrading the performance of a MapReduce job is dynamic slot configuration and job scheduling. J. Dean et al. 2008 discussed MapReduce programming model. The MapReduce model performs operations victimization the map and reduces functions. Map function gets input from user documents. It generates intermediate key/value for reducing function. It additional processes intermediate key/value pairs and provide output key/value pairs. At associate entry level, MapReduce programming model provided the best data processing results. Currently, it must method the massive volume of data. Thus it provides some consequences whereas processing and generating information sets. It takes a lot of execution time for task initialization, task coordination, and task scheduling. Parallel processing might cause inefficient task execution and low resource utilization. J. Polo et al. calculated the map and reduce task completion time dynamically and update it each minute throughout job execution. Task scheduling policy was based on the priority of each job. Priority was estimated based on the concurrent allocation of jobs. The dynamic scheduler is pre-emptive. It affects resource allocation of low priority

jobs. J. Wolf et al. implemented flexible scheduling allocation theme with Hadoop fair scheduler. A primary concern is to optimize scheduling theory metrics, time interval, make span, stretch, and service Level Agreement. They projected penalty operate for measurement of job completion time, epoch scheduling for partitioning time, moldable scheduling for job parallelization, and malleable scheduling for various interval parallelization. Verma et al. projected point in time aware scheduler, referred to as SLO scheduler. The SLO scheduler takes choices of job ordering and slot allocation. This scheduler's primary duty is to maximize the utility operate by implementing the Earliest deadline initial algorithms. It measures how many numbers of slots needed for scheduling the slots dynamically with a selected job deadline. B. Sharma et al. projected a world resource manager for the job tracker and an area resource manager for the task tracker. A world resource manager operate is to manage every MapReduce task. It processes resource needs and resource assignments for every task. a local resource manager's duty is to identify every task. It examines resource usage and task completion time of the task. It deals with detecting bottlenecks with resources and resource contention. Apache Hadoop released next generation MapReduce, referred to as YARN. It replaces MRv1 fixed slot configuration. YARN deals with CPU cores and memory requirements. It splits the job tracker into two components; they are resource managements and job scheduling.

III. FRAME WORK

To maximize the slot utilization for MapReduce and balance the performance exchange between a single job and a batch of jobs with fair scheduling and improving the performance of MapReduce cluster in Hadoop. Goals and Objective the objective is to utilize the slots in MapReduce cluster. The slot utilization remains a challenging task because of fairness and resource needs. It is truthful once all pools are allotted with a similar quantity of resources. The resources needs between the map slot and reduce slot are typically different. This is as a result of the map task and reduce task are often exhibit completely different execution patterns. We tend to review job ordering optimization. To model performance of system, make span and total completion time is used. Total time taken to complete job is calculated. We tend to describe the dynamic slot allocation framework that produces the optimized job order and additionally prove its approximation ratio. We tend to additionally describe the job order which provides the worst, i.e., longest make-span, which is used for derivation of the boundary make-span of a workload. We tend to propose an alternative technique known as dynamic hadoop slot allocation by keeping the slot based model. It relaxes the slot allotment constriction to allow slots to be reallocated to either map or reduce tasks depending on their needs. Second, the speculative execution will tackle the straggler problem that is shown to boost the performance for single job however at the expense of the clustering. Within the view, we tend to propose speculative execution performance balancing to balance performance trade-off between single job and a batch of jobs. Third, delay scheduling has shown to enhance the data vicinity however at the cost of fairness.

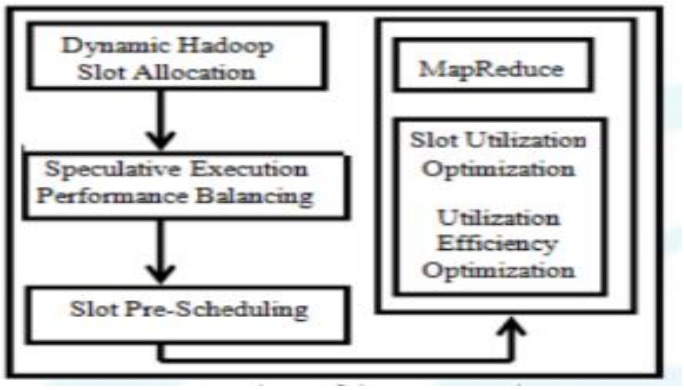


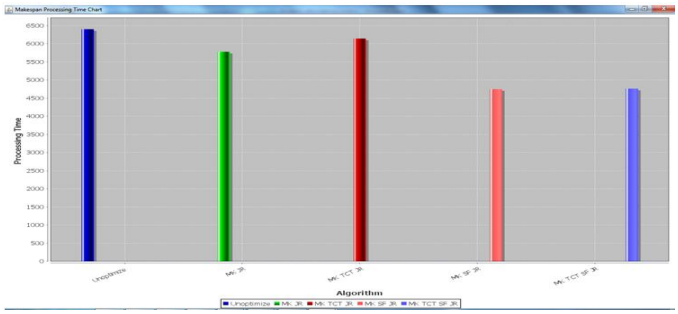
Figure 1: Proposed System Architecture

Finally, by merge these procedures together, we tend to form step by step slot distribution scheme referred to as

Dynamic MR that can improve performance of map reduce workloads well. Programming and Resource Allocation Optimization: Compared to the present development, our projected technique is appropriate for all types of jobs. Echinoderm framework will modify the hadoop configuration mechanically for the MapReduce jobs. By using sampling technique and cost based model we are able to maximize the use of hadoop cluster. However still we are able to improve the performance of this method by maximizing the use of map and by reducing slots. Projected a way for MapReduce multi job workloads based on source conscious indoctrination technique this method concentrate on improving resource utilization by increasing the abstraction of existing task slot to job solve the inefficiency downside of the Hadoop MRv1 within the perspective of resource management. Instead of using slot, it manages resources into containers. The Map and reduce operation are performed on any instrumentality. Speculative Execution Optimization: In MapReduce we need task programming strategy for managing issues such as laggard problem for one job, that include Speculative execution is such an important task programming strategy. The speculative execution algorithm speculates the task by prioritizing and pays attention on heterogeneous environments. To run, choosing the fast nodes and also the speculative tasks are covered over, this speculative execution algorithm may be a longest approximate time to end (LATE) and also the prioritizing of task is needed for speculation. Guo et al. proposes a BASE or Benefit Aware Speculative Execution rule that appraise the probable benefit of the exploratory tasks and also the unnecessary runs are eliminated. This BASE algorithm of the evaluating and elimination will improve the performance for LATE. The speculative execution strategy magnifies its focus primarily on saving cluster computing resource. Maximum cost Performance (MCP) may be a new speculative execution algorithm projected by the projected for fixing the problem that was affecting the performance of the previous speculative execution ways. We tend to projected speculative Execution improvement strategy that balances the tradeoffs between a single job and a group of jobs.

IV. EXPERIMENTAL RESULTS

In our experiments, we are taking the three types of jobs like word count, sorting and creating inverted index after that run the Un Optimized means run the normal map reduce concept it shows the Job ID like serial number Job Name like Word Count, Sorting and Creating Inverted Index, Processing Type like Un Optimized, Processing Time, Mapper Time and Reducer Time the total time taken by Mapper and reducer to process the job is represented as processing time in ms and their individual timings in ns after that apply the MK_JR, MK_TCT_JR and MK_SF_JR algorithms based on that we are maximize the slot utilization for Multiple MapReduce Jobs through Job Ordering Technique. In the below chart we can observe that difference between the lengths of UnOptimized, MK_JR, MK_TCT_JR, MK_SF_JR and MK_TCT_SF_JR Algorithms



We can observe that MakeSpan Processing Time chart in that difference between the lengths of UnOptimized, MK_JR, MK_TCT_JR, MK_SF_JR and MK_TCT_SF_JR Algorithms. The difference will be shown in the sense of Processing Time. Through our implementation we can improve the performance of the system at lower cost then compare to current methods as well as minimize the Makespan and the total completion

time and job ordering optimization for a MapReduce workload under a given map/reduce slot configuration through job ordering technique.

V.CONCLUSION

This paper focuses on the job ordering and map/reduce slot configuration problems for MapReduce production workloads that run periodically during a data warehouse, wherever the typical execution time of map/reduce tasks for a MapReduce job can be profiled from the history run, under the FIFO scheduling during a Hadoop cluster. Two performance metrics are considered, i.e., Makespan and total completion time. We tend to initial specialize in the Makespan. We tend to propose job ordering optimization algorithm and map/reduce slot configuration optimization algorithm. We tend to observe that the entire completion time are often poor subject to obtaining the optimal Makespan, therefore, we tend to additional propose a brand new greedy job ordering algorithm and a map/reduce slot configuration algorithm to minimize the Makespan and total completion time together. The theoretical analysis is additionally given for our projected heuristic algorithms, as well as approximation ratio, higher and lower bounds on Makespan. Finally, we tend to conduct extensive experiments to validate the effectiveness of our projected algorithms and their theoretical results.

REFERENCES

[1] A. Floratou, J. M. Patel, E. J. Shekita, and S. Tata, "Column oriented storage techniques for map reduce," Proc. VLDB Endowment, vol. 4, 419–429, Apr. 2011.

[2] J. Gupta, A. Hariri, and C. Potts, "Scheduling a two-stage hybrid flow shop with parallel machines at the first stage," Ann. Oper. Res., vol. 69, pp. 171–191, 1997.

[3] H. Herodotou and S. Babu, "Profiling, what-if analysis, and cost based optimization of map reduce programs," Proc. Endowment, vol. 4, no. 11, pp. 1111–1122, 2011.

[4] H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F. B. Cetin, and S. Babu, "Starfish: A self-tuning system for big data analytics," in Proc. 5th Conf. Innovative Data Syst. Res., 2011, pp. 261–272.

[5] S. Ibrahim, H. Jin, L. Lu, B. He, and S. Wu, "Adaptive disk I/O scheduling for map reduce in virtualized environment," in Proc. Int. Conf. Parallel Process., Sep. 2011, pp. 335–344.

[6] D. Jiang, B. C. Ooi, L. Shi, and S. Wu, "The performance of map reduce: An in-depth study," Proc. VLDB Endowment, vol. 3, pp. 472–483, Sep. 2010.

[7] S. M. Johnson, "Optimal two- and three-stage production schedules with setup times included," Naval Res. Logistics Quart., vol. 1, no. 1, pp. 61–68, 1954.

[8] H. Karloff, S. Suri, and S. Vassilvitskii, "A model of computation for map reduces," in Proc. ACM-SIAM Symp. Discrete Algorithms, 2010, pp. 938–948.

[9] G. J. Kyparisis and C. Koulamas, "A note on makespan minimization in two-stage flexible flow shops with uniform machines," vol. 175, pp. 1321–1327, 2006.

[10] J. Leung, L. Kelly, and J. H. Anderson, Handbook of Scheduling: Algorithms, Models, and Performance Analysis. Boca Raton, FL, USA: CRC Press, 2004.

[11] B. Moseley, A. Dasgupta, R. Kumar, and T.Sarlos, "On scheduling in map-reduce and flow-shops," in Proc. 23rd Annu. ACM Symp. Parallelism Algorithms Archit., 2011, pp. 289–298.

[12] P.-F. Dutot, L. Eyraud, G. Mounie, and D. Trystram, "Bi-criteria algorithm for scheduling jobs on cluster platforms," in Proc. Archit., 2004, pp. 125–132.

[13] J. Dittrich, J.-A.-Quiane Ruiz, A. Jindal, Y. Kargin, V. Setty, and J. Schad, "adoop++: Making a yellow elephant run like a cheetah," Proc. VLDB Endowment, vol. 3, nos, pp. 515–529, Sep. 2010.

[14] J. N. D. Gupta, "Two-stage, hybrid flowshop scheduling problem," Res. Soc., vol. 39, no. 4, pp. 359–364, 1988.