ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 11, November 2017

Minimizing the Make span and Total Completion Time by Implementing a Novel Greedy Job Algorithm for Map Reduce Production Workloads

[1] D.Anusha, [2] M.Hanimi Reddy
[1][2] CVR College of Engineering/CSE, Hyderabad, India

***Abstract*— The MapReduce is an open supply Hadoop framework applied for processing and generating distributed huge Terabyte facts on big clusters. Its principal motive is to lowering the of completion time of massive units of MapReduce jobs. Hadoop Cluster most effective has predefined constant slot configuration for cluster lifetime. This fixed slot configuration can also produce long completion time (Makespan) and system low resource utilization. The current open source Hadoop allows only static slot configuration, like fixed numbers of map slots and reduce slots throughout the cluster lifetime. Such static configuration may lead to long completion length as well as low system resource utilizations. Propose new schemes which use slot ratio between Map and Reduce tasks as a tunable knob for minimizing the completion length (i.e., makespan) of a given set. By leveraging the workload information of recently completed jobs, schemes dynamically allocates resources (or slots) to map and reduce tasks. Many scheduling methodologies are discussed that aim to improve execution performance as well as completion time goal.**

I. INTRODUCTION

In present years, MapReduce has come to be a famous version for data-extensive computation. The schedulers are important in improving the performance of MapReduce/Hadoop in presence of multiple jobs with specific traits and overall performance goals. In a huge cluster with heterogeneous sources, maximizing a project's distribution can also result in large communication expenses. Therefore, the corresponding job's finishing touch time could be prominent.

MapReduce is a processing method and a software model for dispensed computing based on java. The MapReduce algorithm contains two critical tasks, namely Map and Reduce. Map takes a hard and fast of data and converts it into some other set of data, where man or woman elements are broken down into tuples (key/value pairs). Secondly, lessen undertaking, which takes the output from a Map as an input and combines the ones information tuples right into a smaller set of tuples. As the collection of the name MapReduce implies, the reduce mission is continually carried out after the Map job. MapReduce is that it is straightforward to scale data processing over multiple computing nodes. Under the MapReduce version, the information processing primitives are called mappers and reducers. Decomposing a data processing utility into mappers and reducers is sometimes nontrivial. But, once we write application within the MapReduce form, scaling the software to run over loads, lots, or maybe tens of heaps of machines in a cluster is simply a configuration change. This easy scalability is what has attracted many programmers to use the MapReduce model.

There are two key performance metrics i.e. Makespan and total completion time (TCT). Makespan is defined as the time period since the start of the first job until the completion of the last job for a set of jobs. It considers the computation time of jobs and is often used to measure the performance and utilization efficiency of a system. In contrast, total completion time is referred to as the sum of completed time periods for all jobs since the start of the first job. In this works, we aim to optimize these two metrics. When large data sets are given to operate on them for each or every single set of jobs the makespan and total completion time should be able to minimize and optimize the job reduction issues using MapReduce algorithm. This including the total completion time for all datasets should be optimized. To improve the workload distribution among the multiple machines MapReduce Framework is used. To avoid more memory consumption using efficient technique, MapReduce can be used.

The importance of MapReduce is escalating step by step as a parallel programming model for large scale data preparing. At the same time in the long run, we find out some customary MapReduce stages which have a poor execution in content of group asset use following the conventional multi-stage parallel model and some current

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 11, November 2017

time table approaches utilized as a part of the group environment have a few disadvantages. We address these issues through our involvement in planning a Dynamic Split Model of the assets usage which contains two advances, Dynamic Resource Allocation considering the stage need and employment prerequisite when distributing assets and Resource Usage Pipeline which can relegate errands alertly. The fundamental pipe line of parallel processing catches the scholastic world consideration. Further area/circle is an appropriated figuring stage which is like Google GFS/MapReduce. It comprises of a parallel runtime Sphere and in addition a conveyed file framework Sector. Another parallel runtime is phaser which is a direction build for element parallelism under the environment of multi-processors rather than multi-nodes.

2. RELATED WORK

P.F. Dutot, G.Mounie, and D. Trystram offered an attractive version for scheduling effectively programs on parallel and disbursed systems primarily based on PTs. It is a pleasant opportunity to conventional computational fashions particularly for huge conversation delays and new hierarchical structures. We have proven how to reap properly approximation scheduling algorithms for the distinct styles of PTs for two standards for both off-line and on-line instances.

J. N. D. Gupta described that the two-level flow shop problem while there are same a couple of machines at each level, and shows that the problem is NP-complete. An efficient heuristic algorithm is developed for finding an approximate answer of a unique case whilst there may be most effective one machine at stage2.

Anil Sagar T, Ramakrishna V Moni proposed a Dynamic MR Technique can be used to beautify the execution of MapReduce workloads even as retaining up the fairness. It incorporates of 3 strategies, mainly, DHSA, SEPB, Slot PreScheduling, all of which focus on the slot use optimization for MapReduce group from trade points of view. DHSA concentrates on the slot use enlargement by way of dispensing map or reduce slots to map and reduce obligations alterably. Especially, it doesn't have any presumption or require any earlier studying and can be applied for any kinds of MapReduce jobs (e.G., self reliant or subordinate ones). Two varieties of DHSA are brought, particularly, PI-DHSA and PD-DHSA, in view of exceptional levels of fairness. Client can choose each of them likewise. Slot Pre-Scheduling complements the talent of slot use through using growing its statistics locality.

S. R. Hejazi and S. Saghafian, survey of flow shop scheduling problems and contributions from early works of Johnson of 1954 Johnson, SM. 1954. Optimal two- and three-stage production schedules with set-up times included. Naval Res. Logist. Quart., 1: 61–68. to recent approaches of met heuristics of 2004. It mainly considers a flow shop problem with a make span criterion and it surveys some exact methods (for small size problems), constructive heuristics and developed improving met heuristic and evolutionary approaches as well as some well-known properties and rules for this problem. Each part has a brief literature review of the contributions and a glimpse of that approach before discussing the implementation for a flow shop problem. Moreover, in the first section, a complete literature review of flow shop related scheduling problems with different assumptions as well as contributions in solving these other aspects is considered. This paper can be seen as a reference to past contributions (particularly in n/m/p/c max or equivalently F/prmu/c max) for future research needs of improving and developing better approaches to flow shop-related scheduling problems.

C. Oguz, and M. F. Ercan developed several efficient heuristic algorithms to schedule unit processing time multiprocessor tasks in a two-stage hybrid flow-shop for minimizing makespan. They also derived two effective lower bounds for the problem. Then, they analyzed the average performance of the heuristic algorithms by computing the average relative gap of each heuristic solution from the lower bound. The results of the computational experiment to test the average performance of the proposed heuristic algorithms on a set of randomly generated problems showed that three of the proposed heuristic algorithms perform well.

A. Rasmussen, V. T. Lam, M. Conley, G. Porter, R. Kapoor, and A.Vahdat present Themis, a MapReduce implementation that reads and writes data records to disk exactly twice, which is the minimum amount possible for

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 11, November 2017

data sets that cannot fit in memory. In order to minimize I/O, Themis makes fundamentally different design decisions from previous MapReduce imple mentations. Them is performs a wide variety of Map Reduce jobs including click log analysis, DNA read sequence alignment, and Page Rank at nearly the speed of Triton Sort's record-setting sort performance.

S.Tang, B.-S. Lee, and B.He explained MapReduce is a popular parallel computing paradigm for large-scale data processing in clusters and data centers. However, the slot utilization can be low, especially when Hadoop Fair Scheduler is used, due to the pre-allocation of slots among map and reduce tasks, and the order that map tasks followed by reduce tasks in a typical MapReduce environment. To address this problem, we propose to allow slots to be dynamically (re)allocated to either map or reduce tasks depending on their actual requirement. Specifically, we have proposed two types of Dynamic Hadoop Fair Scheduler (DHFS), for two different levels of fairness (i.e., cluster and pool level). Th e experimental results show that the proposed DHFS can improve the system performance significantly (by 32% 55% for a single job and 44% 68% for multiple jobs) while guaranteeing the fairness.

H.Herodotou, H. Lim, G. Luo, N. Borisov, explained Timely and cost-effective analytics over "Big Data" is now a key ingredient for success in many businesses, scientific and engineering disciplines, and government endeavors. The Hadoop software stack which consists of an extensible Map Reduce execution engine, pluggable distributed storage engines, and a range of procedural to declarative interfaces is a popular choice for big data analytics. Most practitioners of big data analytics—like computational scientists, systems researchers, and business analysts lack the expertise to tune the system to get good performance. Unfortunately, Hadoop's performance out of the box leaves much to be desired, leading to suboptimal use of resources, time, and money (in pay- as-you-go clouds). We introduce Starfish, a self-tuning system for big data analytics. Starfish builds on Hadoop while adapting to user needs and system workloads to provide good performance automatically, without any need for users to understand and manipulate the many tuning knobs in Hadoop. While Starfish's system architecture is guided by work on self-tuning database systems, we discuss how new analysis practices over big data pose new challenges; leading us to different design choices in Starfish

Chen He, Ying Lu and David Swanson developed a new matchmaking algorithm to improve the data locality rate and the average response time of MapReduce clusters. We have carried out experiments to compare not only MapReduce scheduling algorithms with and without our matchmaking algorithm but also with an existing data locality enhancement technique.

Dawei Jiang, Beng Chin Ooi, Lei Shi and Sai Wu conducted an in-depth performance study of MapReduce in its open source implementation, Hadoop. We have identified five factors that affect the performance of MapReduce and investigated alternative implementation strategies for each factor.

S. Ibrahim, H. Jin, L. Lu, B. He, and S. Wu Virtual machine (VM) interference has long been a challenging problem for performance predictability and system throughput for large-scale virtualized environments in the cloud. Such interferences are contributed by intertwined factors including the application's type, the number of con current VMs, and the VM scheduling algorithms used within the host. Since Map Reduce has become an important data processing platform in the cloud, they investigate the impact of disk schedulers in Hadoop. Interestingly, our experimental results report a noticeable variation of the Hadoop performance between different applications when applying different disk pairs' schedulers in both the hypervisor and the virtual machines. Furthermore, a typical Hadoop application consists of different interleaving stages, each requiring different I/O workloads and patterns. As a result, the disk pairs' schedulers are not only sub-optimal for different Map Reduce applications, but also sub optimal for different sub-phases of the whole job. Accordingly, they presents a novel approach for adaptively tuning the disk pairs' schedulers in both the hypervisor and the virtual machines during the execution of a single Map Reduce job. Our results show that Map Reduce performance can be significantly improved; specifically, adaptive tuning of disk pairs' schedulers achieves a 25% performance improvement on a sort benchmark with Hadoop.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 11, November 2017

T. Nykiel, M. Potamias, C. Mishra, G. Kollios, and N. Koudas survey on Large-scale data analysis lies in the core of modern enterprises and scientific research. With the emergence of cloud computing, the use of an analytical query processing infrastructure (e.g., Amazon EC2) can be directly mapped to monetary value. Map Reduce has been a popular frame work in the context of cloud computing, designed to serve long running queries (jobs) which can be processed in batch mode. Taking into account that different jobs often perform similar work, there are many opportunities for sharing. In principle, sharing similar work reduces the overall amount of work, which can lead to reducing monetary charges incurred while utilizing the processing infrastructure. They propose a sharing framework tailored to Map Reduce. Our framework, MR Share , transforms a batch of queries into a new batch that will be executed more efficiently, by merging jobs into groups and evaluating each group as a single query. Based on our cost model for Map Reduce, they define an optimization problem and we provide a solution that derives the optimal grouping of queries. Experiments in our prototype, built on top of Hadoop, demonstrate the overall effectiveness of our approach and substantial savings.

3. FRAMEWORK

A. Proposed System Overview

We goal at one subset of manufacturing MapReduce workloads that consist of a set of unbiased jobs (e.g., every of jobs approaches distinct information units and not using a dependency among each different) with one-of-a-kind procedures. For dependent jobs (i.e., MapReduce workflow), one MapReduce can most effective begin simplest whilst its preceding structured jobs end the computation concern to the input-output statistics dependency. In comparison, for independent jobs, there's an overlap computation among two jobs, i.e., while the current job completes its map section computation and starts its map phase computation, the subsequent task can begin to perform its map-segment computation in a pipeline processing mode by means of possessing the launched map slots from its preceding activity.

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.

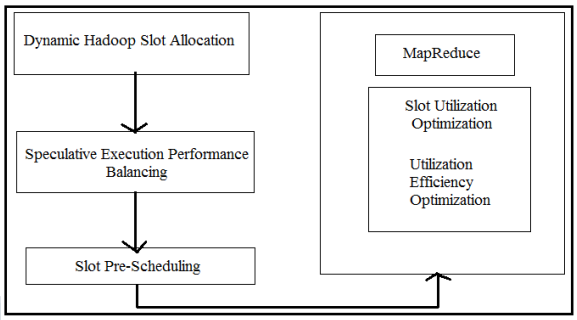


Fig1. Overview of Framework

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

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 11, November 2017

shown to enhance the data vicinity however at the cost of fairness.

B. Need for Makespan Optimization

In this paper, Makespan and Total Completion Time (TCT) are two major performance metrics to the MapReduce workloads. Generally, makespan is defined because the term for the reason that begin of the first task until the entirety of the final task for a hard and fast of jobs. It considers the computation time of jobs and is often used to degree the overall performance and utilization efficiency of a machine. In assessment, general of completion time is called the sum of finished time periods for all jobs for the reason that begin of the primary job. It is a generalized makespan with queuing time covered. We can use it to measure the satisfaction to the device from a single task's perspective thru dividing the entire of entirety time by using the quantity of jobs.

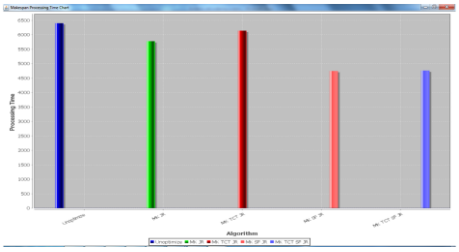
In this paper, we describe the MK_JR set of rules that produces the optimized task order and also prove its approximation ratio. We also describe the process order which offers the worst, i.e., longest makespan, that's used for derivation of the higher sure makespan of a workload. Next, we describe the MK_TCT_JR algorithm, which optimizes each makespan and overall of completion time.

4. EXPERIMENTAL RESULTS

In this experiment, we need run unoptimized (by using normal map reducer concept). Here we are running 3 types of jobs such as word count, sorting and creating inverted index. After run these three jobs, we can run the MK_JR algorithm. As per this algorithm, we order jobs in J based on the following principles: Partition jobs set J into two disjoint sub-sets JobA and JobB:

JobA = when T(m) <= T(r)

JobB = when T(m) > T(r)



MK_TCT_JR algorithm is also similar to MK_JR algorithm but difference is based on time threshold value it arrange the jobs. After completion of these two algorithms, we can run the MK_SF_JR algorithm. This algorithm shows that how many processes required for job and MK_TCT_SF_JR algorithm provide the processes information based on time threshold. Finally, we can see the makespan time for all algorithms.

5. CONCLUSION

In this paper we worked on the job ordering and map/reduce slot configuration problems for MapReduce production workloads, wherever the typical execution time of map/reduce tasks for a MapReduce job can be profiled from the history run. 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] S. R. Hejazi and S. Saghafian, "Flowshop-scheduling problemswith makespan criterion: A review," Int. J. Production Res., vol. 43,no. 14, pp. 2895–2929, 2005.

[2] S. Agarwal, S. Kandula, N. Bruno, M.-C. Wu, I. Stoica, and J.Zhou, "Re-optimizing data-parallel computing," in Proc. 9th USENIX Conf. Netw. Syst. Design Implementation, 2012, p. 21.

[3] P. Agrawal, D. Kifer, and C. Olston, "Scheduling shared scans oflarge data files," Proc. VLDB Endow., vol. 1, no. 1, pp. 958–969,Aug. 2008.

[4] W. Cirne and F. Berman, "When the herd is smart: Aggregatebehavior in the selection of job request," IEEE Trans. Parallel Distrib. Syst., vol. 14, no. 2, pp. 181–192, Feb. 2003.

[5] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, K. Elmeleegy,and R. Sears, "Mapreduce online," in Proc. 7th USENIX Conf.Netw. Syst. Design Implementation, 2010, p. 21.

[6] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," in Proc. 6th Conf. Symp. Oper. Syst. DesignImplementation, 2004, vol. 6, p. 10.

[7] J. Dittrich, J.-A.-Quiane Ruiz, A. Jindal, Y. Kargin, V. Setty, and J.Schad, "adoop++: Making a yellow elephant run like a cheetah(without it even noticing)," Proc. VLDB Endowment, vol. 3,nos. 1–2, pp. 515–529, Sep. 2010.

[8] P.-F. Dutot, L. Eyraud, G. Mounie, and D. Trystram, "Bi-criteriaalgorithm for scheduling jobs on cluster platforms," in Proc. 16thAnnu. ACM Symp. Parallelism Algorithms Archit., 2004, pp. 125–132.

[9] P.-F. Dutot, G.Mounie, and D. Trystram,"Scheduling paralleltasks: Approximation algorithms," in Handbo ok of Scheduling:Algorithms, Models, and Performance Analysis, J. T. Leung, Ed. BocaRaton, FL, USA: CRC Press, ch. 26, pp. 26-1–26-24.

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

ABOUT AUTHORS:

D. Anusha is currently pursuing her M.Tech in Computer Science and Engineering, CVR College of Engineering, Hyderabad, Telangana. She received her B.E in Computer Science and Engineering Department from Sphoorthy engineering college, Hyderabad.

M. Hanimi Reddy is currently working as an Assistant Professor in Computer Science and Engineering Department, CVR Engineering College, Hyderabad.