ISSN: 2278-4632
Vol-10 Issue-7 No. 11 July 2020

DYNAMIC JOB SCHEDULING FOR MAPREDUCE WORKLOADS THROUGH SLOT CONFIGURATION TECHNIQUE

Mr. G. J. SUNNYDEOL

Assistant Professor, Department of CSE, Sri Mittapalli College of Engineering, Tummalapalem, on NH-16, Guntur, Andhra Pradesh, India.

1) O. SAI, 2) R. SANKAR, 3) A. AJAY, 4) K. NAVEEN

B. Tech Student, Department of CSE, Sri Mittapalli College of Engineering, Tummalapalem, on NH-5, Guntur, Andhra Pradesh, India.

Abstract: MapReduce is a popular parallel computing paradigm for large-scale data processing in clusters and data centers. A MapReduce workload generally contains a set of jobs, each of which consists of multiple map tasks followed by multiple reducetasks.

Due to 1) that map tasks can only run in map slots and reduce tasks can only run in reduce slots, and 2) the general executionconstraints that map tasks are executed before reduce tasks, different job execution orders and map/reduce slot configurations for a MapReduce workload have significantly different performance and system utilization. This paper proposes two classes of algorithms to minimize the makespan and the total completion time for an offline MapReduce workload. Our first class of algorithms focuses on the job ordering optimization for a MapReduce workload under a given map/reduce slot configuration. In contrast, our second class of algorithms considers the scenario that we can perform optimization for map/reduce slot configuration for a MapReduce workload.We perform simulations as well as experiments on Amazon EC2 and show that our proposed algorithms produce results that are up to 50- 80 percent better than currently unoptimized Hadoop, leading to significant reductions in running time inpractice.

I. INTRODUCTION

MAPREDUCE is a widely used computing model for large scale data processing in cloud computing. A MapReduce job consists of a set of map and reduce tasks, where reduce tasks are performed after the maptasks.Hadoop, an open source implementation of MapReduce, has been deployed in large clusters containing thousands of machines by companies such as Amazon and Facebook. In those cluster and data center environments, MapReduce and Hadoop are used to support batch processing for jobs submitted from multiple users (i.e., MapReduce workloads).Despite many research efforts devoted to improve the performance of a single MapReduce job, there is relatively little attention paid to the system performance of MapReduce workloads. Therefore, this paper tries to improve the performance of Map Reduce work loads .Makespan and total completion time (TCT) are two key performance metrics. Generally, 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. It is a generalized makespan with queuing time (i.e., waiting time) included.We can use it to measure the satisfaction to the system from a single job's perspective through dividing the total completion time by the number of jobs (i.e., average completion time).

Therefore, in this paper, we aim to optimize these two metrics.

II. METHODOLOGY

In this section, we give an overview of related work from two aspects. First, we review batch job ordering optimization work in HPC literature. Second, we summarize the MapReduce job optimization work proposedin recentyears.

A. Job Ordering Optimization

The batch job ordering problem has been extensively studied in the high performance computing literature. Minimizing the makespan has been shown to be NP-hard, and a number of approximation and heuristic algorithms have been proposed.

ISSN: 2278-4632
Vol-10 Issue-7 No. 11 July 2020

In addition, there has been work on bi-criteria optimization which aims to mini size makespan and total completion time simultaneously, such as.

The previous works all focused on the single-stage parallelism, where each job only has a single stage. In contrast, MapReduce is an interleaved parallel and sequential computation model which is related to the two-stage hybrid flow shop (2HFS) problem. Minimizing the makespan for 2HFS is strongly NP-hard when at least one stage contains multiple processors. There has been a large body of approximation and heuristic algorithms proposed for 2HFS. Additionally, there has been work targeted at the bi-criteria optimization of both make- span and total completion time.

The main difference between MapReduce and traditional 2HFS is that MapReduce jobs can run multiple map and reduce tasks concurrently in each phase, whereas 2HFS allows at most one task to be processed at a time. In this way, MapReduce is more similar to the two-stage hybrid flow shop with multiprocessor tasks (2HFSMT), problem, which allows a task at each stage can be processed on multiple processorssimultaneously. One is a greedy algorithm job ordering method based on Johnson's Rule.Another is a heuristic algorithm called Balanced Pool. They discuss and evaluate the algorithms experimentally. We follow their job ordering approach (i.e., MK_JR algorithm in our paper). But our main contributions go beyond it in a number of significant aspects.First, we prove a 1 þ d upper bound on the approximation ratio of our MK_JR algorithm.

Second, we give the relationship between upper-bound makespan, lower-bound makespan, and the corresponding job orders. Additionally, our MK_TCT_JR algorithm obtains a trade-off in the makespan and total completion time, which produces very good results. Moreover, for online workloads, we proposed a prototype named MROrder to perform online job ordering optimization by incorporating MK_JRalgorithm.

B. MapReduce Job Optimization

There is a large body of research work that focuses on the optimization for MapReduce jobs. One optimization policy focuses on the architectural design and optimization issues.

We propose a set of general low-level optimizations including improving I/O speed,utilizing indexes, using fingerprinting for faster key comparisons, and block size tuning. presented an I/O-efficient MapReduce system called Themis that improves the performance of MapReduce by minimizing the number of I/O operations. Likewise, Sailfish improves MapReduce's performance through more efficient disk I/O. It mitigates partitioning skew in MapReduce by choosing the number of reduce tasks and intermediate data partitioning dynamically at runtime, using an index constructed on intermediate data. There are also methods that reduce I/O cost in MapReduce by using indexing structures, column-oriented storage. proposed a scheduling technique and implemented a prototype called Adaptive Scheduler that can adaptively manage the workload performance with the awareness of hardware heterogeneity, distributed storage to meet user's deadline requirement. propose a flexible scheduling allocation scheme called FLEX, which can optimize any of a variety of standard scheduling theory metrics, such as response time, stretch, makespan. proposed a dynamic slot allocation system called DynamicMR to improve the performance for the slot-based Hadoop MRv1, by allowing map (or reduce) tasks can be run on map slots and reduceslots.

C. Problem FormulationAnd PerformanceModel

In this section, we give a formal model for MapReduce and formalize its associated optimization problems.

1) ProblemFormulation

A MapReduce job Ji computation consists of two phases, a map phase M and reduce phase R. Each phase consists of a number of tasks. We writejJiMjandjJiRjforthenumberoftasksin

Ji's map phase and reduce phase, respectively. Let tMi;j and tRi;j denote the execution time of Ji'sjjth map task and jth reduce task, respectively. We consider a MapReduce workload with a set of independent jobs J ¼ {J1; J2; .. . Jn}, for some n. These jobs can be executed in any order. The workload is executed on a MapReduce cluster under FIFO scheduling, consisting of a set of (map and reduce) slots, denoted as S.

ISSN: 2278-4632
Vol-10 Issue-7 No. 11 July 2020

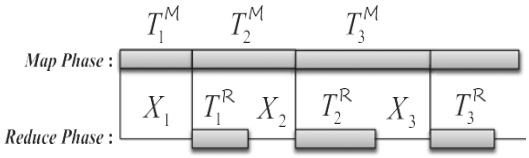

Fig: Flow for the simplified case MapReduce tasks execution

Let f denote the job submission order for a MapReduce workload. We focus on the offline situation in which all the jobs are available at time 0. Let ci denote the completion time of Ji (i.e., the time when Ji's reduce tasks all finish). The makespan for the workload J1; ... ; Jn is defined as Cmax ¼ maxi2½n]fcig. The total completion time for the workload is defined as Ctct ¼Pi2½n]ci.

In our work, we consider four optimization problems, defined as follows:

Problem 1. Find an ordering f to execute the jobs J1; ... ; Jn in a MapReduce workload such that Cmax is minimized, under a given slot configuration ðSM; SRÞ?

Problem 2. Find an ordering f to execute the jobs J1; ... ; Jn in a MapReduce workload that can optimize (minimize) Cmax and Ctct simultaneously, under a given slot configuration ðSM; SRÞ?

Moreover, if we are MapReduce cluster administrators, we can perform the following optimization work:

Problem 3. Find a map/reduce slot configuration ðSM; SRÞ and ordering f to execute the jobs J1;
... ; Jn in a MapReduce workload such that Cmax is minimized, under a given value of total slotsS?

Problem 4. Find a map/reduce slot configuration ðSM; SRÞ and ordering f to execute the jobs J1;
Jn in a MapReduce workload that can optimize (minimize) Cmax and Ctctsimulta- neously, under a given value of total slots S?

2)Performance Model for Makespan and Total CompletionTime

In this section, we aim to deduce the mathematical per- formancemodel formakespan and total completion time. We start by considering a simplified case where we can give a close-form formula for makespan and total completion time.

Next, we consider the general case in which it is complex and difficult to get the exact mathematical formula. Instead, we deduce an upper bound forit.

We first consider a simplified case where jSMj¼ 1 andjSRj¼ 1. It turns out to be a perfect two-machine flow-shop problem. gives an example of an execution for this case. For each job Ji, let TiM be the total processing time of map tasks and TiR be the total processing time for the reduce tasks. Let Xi be the idle period of time for reduce machines before the reduce tasks of job Ji startrunning.

D. Job Ordering Optimization ForMapreduce Workload

his section attempts to address Problem 1 and Problem 2. We first focus on makespan optimization. We describe the MK_JR algorithm that produces the optimized job order and also prove its approximation ratio. We also describe the job order which gives the worst, i.e., longest makespan, which is used for derivation of the upper bound makespan of a workload. Next, we describe the MK_TCT_JR algorithm, which optimizes both makespan and total completion time. Finally, it shows that the orderings produced by MK_JR and MK_TCT_JR are stable, even when MapReduce servers fail.

1) Makespan Optimization

Recall the simplified case described in the previous section, where the number of map and reduce tasks of all the jobs were divisible by the number of map and reduce slots. The optimal job order for the simplified case can be obtained by using Johnson's Rule which is an efficient job ordering algorithm for the minimum makespan Copt max for the two-stage flow shop with one processor per stage. When the number of tasks is not divisible by the number of slots, the makespan minimization problem becomes NP-hard which has first noted it and proposed an algorithm based on Johnson's rule.

2) Bi-Criteria Optimization of Makespan and Total Completion Time

Makespan and total completion time are two key performance metrics. Generally, makespan refers to the maximum completion time for a batch of jobs.

It considers the computation time of jobs and is

ISSN: 2278-4632
Vol-10 Issue-7 No. 11 July 2020

often used to measure the performance and utilization efficiency of a system.

In contrast, total completion time is the sum of completion time of all jobs. It is a generalized makespan with queuing time (i.e., waiting time) included. It can be used to measure the satisfaction to the system from a single job's perspective. So far, we focus only on the optimization of makespan. Note that the total completion time that can be poor subject to obtaining optimal makespan, as illustrated in Fig. Therefore, there is a need for bi-criteria optimization on both makespan and total completion time. Intuitively, the makespan is affected primarily by the positions of large-size jobs. In contrast, the total completion time is mainly influenced by the positions of small-size jobs. The algorithm shortest processing time first (SPTF) is optimal for the total completion time on a single machine where there is one task per job and no precedence constrains.

However, MK_JR is not aware of varying job sizes. Indeed, the job order produced by MK_JR can have adverse effect on the total completion time if we follow Johnson's Rule strictly in some scenarios.

III. EXPERIMENTRESULTS

To well reflect practical workloads, we generate our testbed workloads by choosing nine benchmarks arbitrarily from Pur- due MapReduce Benchmarks Suite1 and using their provided datasets. The detailed benchmarks are described asfollows.

WordCount: Computes the occurrence frequency of each word in adocument.Sort: Sorts the data in the input files in a dictionaryorder.

Grep: Finds the matches of a regex in the inputfiles.

InvertedIndex: Takes a list of documents as input and generates word-to-documentindexing.Classification: Classifies the input into one of k pre- determinedclusters.
HistogramMovies: Generates a histogram of input data and is a generic tool used in many dataanalyses.
HistogramRatings: Generates a histogram of the ratings as opposed to that of the movies based on their averageratings.
SequenceCount: Generates a count of all unique sets of three consecutive words per document in the inputdata.
TeraSort: Sorts 100-byte <key,value> tuples on the keys where key is a 10-byte field and the rest of the bytes as value(payload).

A. Job Ordering Optimization Algorithms

Let's begin with the evaluation of job ordering optimization algorithms MK_JR and MK_TCT_JR first presents thenormalized performance results for testbed workloads under three different job orders, i.e., an unoptimized job order based on Theorem 2, the job order based on MK_JR and the job order based on MK_TCT_JR, in a Hadoop cluster, where we configure three map and one reduce slots per slave node. Therefore, we have jSMj¼ 57 and jSRj ¼ 19. For makespan (or total completion time), we normalize it by using makespan speedup (or total completion time speedup), defined as the ratio of makespan (or total completion time) from the unoptimized case to that from the designated job order. Moreover, there is a slight drop in makespan speedup for MK_TCT_JR in comparison to MK_JR, sacrificing a bit performance improvement in makespan for a good total com- pletion time. It can be noted in that MK_TCT_JR has a good total completion time speedup.

B. Slot Configuration Optimization Algorithms

In this section, let's come to evaluate map/reduce slot con- figuration optimization algorithms, namely, Algorithm MK_SF_JR and Algorithm MK_TCT_SF_JR. Fig. 6 illustrates experimental results for testbed workloads under various slot configuration optimization algorithms. Particularly, we take the Hadoop default configuration which sets each slave node with two map slots and two reduce slots as the unoptimized case. Then for 19 slave nodes, it holds jSMj¼ 38 and jSRj¼ 38. Here, there is about 24 ~ 41 percent performance improvement from Algorithm MK_SF_JR. For Algorithm MK_TCT_SF_JR, in contrast, it is a bi-criteria optimization algorithm for make- span and total completion time. They illustrate that there is a significant performance improvement (112 ~ 132) percent for total completion time,

ISSN: 2278-4632
Vol-10 Issue-7 No. 11 July 2020

at the expense of a small drop in makespan improvement in comparison to Algorithm MK_SF_JR.

Likewise, we will show for Facebook workloads that therecan also be a very seriously negativeimpact in total completion time for Algorithm MK_SF_JR, whereas Algorithm MK_TCT_SF_JR can overcome and improve it significantly.

C.Simulation Result with Synthetic Facebook Workload

MapReduce jobs in production at Facebook in October 2009, provided by Zaharia. They are classified into nine bins based on job sizes (numbers of maps). We make our synthetic workloads (SW for short) by picking representative sizes and number of jobs from each bin based on the percentage of the total number of jobs as well as the size of SW.

Table: Facebook and Sizes and Number of Jobs.

Bin	#Maps	% at Facebook	Size in SW	#Jobs in SW
0	1-25	58%	1-25	29
1	25-50	9.6%	25, 30, 35, 40, 50	5
2	50-100	8.6%	60, 80, 90, 100	4
3	100-200	8.4%	120, 150, 180, 200	4
4	200-400	5.6%	250, 320, 400	3
5	400-800	4.3%	600, 800	2
6	800-1600	2.5%	1,200	1
7	1,600-3,200	1.3%	2,400	1
8	>3,200	1.7%	4,800	1

IV. CONCLUSION &FUTUREWORK

This paper focuses on the job ordering and map/reduce slot configuration issues for MapReduce production workloads that run periodically in a data warehouse, where the aver- age execution time of map/reduce tasks for a MapReduce job can be profiled from the history run, under the FIFO scheduling in a Hadoop cluster. Two performance metrics are considered, i.e., makespan and total completion time. We first focus on the makespan.

Map/reduce slot configurationoptimization algorithm. We observe that the total completion time can be poor subject to getting the optimal make-span, therefore, we further propose a new greedy job ordering algorithm to minimize the makespan and total completion time together.

The theoretical analysis is also given for our proposed heuristic algorithms, including approximation ratio, upper and lower bounds on makespan.

Finally, we conduct extensive experiments to validate the effectiveness of our proposed algorithms and their theoretical results.

V. REFERENCES

[1] Amazon ec2 [Online]. Available:http://aws.amazon.com/ec2,2015.

[2]Apache hadoop [Online]. Available:http://hadoop.apache.org, 2015.

[3]Howmanymapsandreduces[Online].

Available:http://wikiapache.org/hadoop/HowMany MapsAndReduces, 2014.

[4] Lognormal distribution [Online]. Available: http://en.wikipedia.

org/wiki/Log-normal_distribution, 2015.

[5] The scheduling problem [Online]. Available: http://riot.ieor.berkeley.edu/Applications/Schedu ling/algorithms.html,1999.

[6]S. R. Hejazi and S. Saghafian, "Flowshop-scheduling problems with makespan criterion: A review," Int. J. Production Res., vol. 43, no. 14, pp. 2895–2929, 2005.

[7]S. Agarwal, S. Kandula, N. Bruno, M.-C. Wu,

I. Stoica, and J. Zhou, "Re-optimizing data-parallel computing," in Proc. 9th USE- NIX Conf. Netw. Syst. Design Implementation, 2012, p. 21.

[8]P. Agrawal, D. Kifer, and C. Olston, "Scheduling shared scans of large data files," Proc. VLDB Endow., vol. 1, no. 1, pp. 958–969,

Aug. 2008.

[9]W. Cirne and F. Berman, "When the herd is smart: Aggregate behavior in the selection of job request," IEEE Trans. Parallel Dis- trib. Syst., vol. 14, no. 2, pp. 181–192, Feb. 2003.

[10]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.

[11]J. Dean and S. Ghemawat, "Mapreduce: Simplified data process- ing on large clusters," in Proc. 6th Conf. Symp. Oper. Syst. Design Implementation, 2004, vol. 6, p.1.

