Future Generation Computer Systems 43–44 (2015) 51–60

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

A self-adaptive scheduling algorithm for reduce start time

Zhuo Tang [a,*], Lingang Jiang [a], Junqing Zhou [a], Kenli Li [a], Keqin Li [a,b]

[a] College of Information Science and Engineering, Hunan University, Changsha 410082, China
[b] Department of Computer Science, State University of New York, New Paltz, NY 12561, USA

HIGHLIGHTS

- This paper illustrates the reasons of the system slots waster for reduces tasks waiting around.
- The model can determine the start time of reduce tasks, dynamically according to job context.
- As an optimal scheduling algorithm, SARS can decrease the reduce completion time for jobs.

ARTICLE INFO

Article history:
Received 28 December 2013
Received in revised form
1 August 2014
Accepted 15 August 2014
Available online 25 August 2014

Keywords:
Big data
Hadoop
MapReduce
Reduce
Self-adaptive
Task scheduling

ABSTRACT

MapReduce is by far one of the most successful realizations of large-scale data-intensive cloud computing platforms. When to start the reduce tasks is one of the key problems to advance the MapReduce performance. The existing implementations may result in a block of reduce tasks. When the output of map tasks become large, the performance of a MapReduce scheduling algorithm will be influenced seriously. Through analysis for the current MapReduce scheduling mechanism, this paper illustrates the reasons of system slot resources waste, which results in the reduce tasks waiting around, and proposes an optimal reduce scheduling policy called SARS (Self Adaptive Reduce Scheduling) for reduce tasks' start times in the Hadoop platform. It can decide the start time point of each reduce task dynamically according to each job context, including the task completion time and the size of map output. Through estimating job completion time, reduce completion time, and system average response time, the experimental results illustrate that, when comparing with other algorithms, the reduce completion time is decreased sharply. It is also proved that the average response time is decreased by 11% to 29%, when the SARS algorithm is applied to the traditional job scheduling algorithms FIFO, FairScheduler, and CapacityScheduler.

1. Introduction

MapReduce is an excellent model for distributed computing, introduced by Google in 2004 [1]. It has emerged as an important and widely used programming model for distributed and parallel computing, due to its ease of use, generality, and scalability. Among its open source implementation versions, Hadoop has been widely used in industry around the whole world [2] and has been used/extended by scientists as the base of their own research work [3]. It has been applied to the production environments, such as Google, Yahoo, Amazon, Facebook, and so on. Because of the short development time, Hadoop can be improved in many aspects, such as intermediate data management and reduce tasks scheduling [4]. This paper mainly focuses on the reduce scheduling problem, which refers to the starting times of the reduce tasks.

Map and Reduce are the two sections in a MapReduce scheduling algorithm. In Hadoop, each task contains three functioning phases: copy, sort, and reduce [5]. The goal of the copy phase is to read the map tasks' outputs. The sort phase is to sort the intermediate data which are produced by map tasks and will be the input to the reduce phase. Finally, the eventual results are produced through the reduce phase, where the copy and sort phases are to preprocess the input data of reduce. In real applications, copying and sorting may consume considerable amount of time, especially in the copy phase. In the theoretical model, the reduce functions start only if all map tasks are finished [6]. However, in the Hadoop implementation, all copy actions of reduce tasks will start when the first map action is finished [7]. But in a slot duration, if there is any map task still running, the copy actions will wait around. This will lead to the waste of the reduce slot resources.

The existing MapReduce program frameworks often treat the jobs as a whole process. However, the differences between the map and reduce tasks are not considered. Since map and reduce task execution times are not related, it is not accurate to compute the

* Corresponding author. Tel.: +86 18627568501.
E-mail address: ztang@hnu.edu.cn (Z. Tang).

http://dx.doi.org/10.1016/j.future.2014.08.011
0167-739X/

average task execution time by taking map and reduce tasks together. The dynamic proportional scheduler [8] provides more job sharing and prioritization capability in scheduling and also results in increasing share of cluster resources and more differentiation in service levels of different jobs. Zaharia et al. proposed the delay scheduling algorithm [9] to address the conflict between data locality and fairness. Time estimation and optimization for Hadoop jobs have been explored in [10,11]. In [10], the authors focused on minimizing the total completion time of a set of MapReduce jobs. In [11], the authors estimated the progress of queries that run as MapReduce DAGs.

The need for a scheduling algorithm arises from the requirement for most modern systems to perform multitasking (execute more than one process at a time) and multiplexing (transmit multiple flows simultaneously) [12]. In traditional MapReduce scheduling, the reduce task should begin when all the map tasks are completed. In this way, the outputs of map tasks should be read and written to the reduce tasks in the copy process [13]. However, through the analysis of the slot resource usage in the reduce process, this paper illustrates that data transfer will result in slot idle and delay. On the other hand, various types of information and data processed in the large-scale dynamic grid environment may be incomplete, imprecise, fragmentary and overloading [14]. So as Hadoop platform in cloud computing, when the map tasks' outputs become large, the performance of a MapReduce scheduling algorithm will be influenced seriously [15]. Especially, when multiple tasks are running simultaneously, inappropriate scheduling of reduce tasks will lead to untimely scheduling of other jobs in the system, and these are all the stumbling blocks of the Hadoop popularization.

Through studying reduce task scheduling in the Hadoop platform, this paper proposes an optimal reduce scheduling policy called SARS (Self Adaptive Reduce Scheduling). SARS can reduce the waiting around of copy actions and advance the performance of a whole system. Through analyzing the details of a map and reduce two-phase scheduling process at runtime, the SARS algorithm can determine the start time point of each reduce task dynamically according to each job's context, such as the task completion time or the size of map output. This paper makes the following contributions to MapReduce performance enhancement:

(1) The analysis for the current MapReduce scheduling mechanism, and illustration of the reasons of system slot resources wasting, which results in the reduce tasks waiting around;
(2) The development of a model details the start times of the reduce tasks dynamically according to each job context, including the task completion time and the size of map output;
(3) An optimal reduce scheduling algorithm which decreases the reduce completion time and the system average response time in the Hadoop platform.

The rest of this paper is organized as follows. Section 2 reviews the related works. Section 3 analyzes the problem of the long waiting of reduce tasks. Section 4 proposes an optimal reduce scheduling algorithm for reduce tasks' start times in the Hadoop platform. Experiments and analysis which support our contributions are presented in Section 5. Section 6 concludes this paper and describes the future work.

2. Related work

There have been a number of proposals for task scheduling in distributed systems, which use various mathematical techniques to achieve the MapReduce scheduling process. At present, the researches on MapReduce scheduling algorithms focus on the optimization of the job computation time, cluster workloads, and data communication.

Balanced-pools efficiently utilize performance properties of MapReduce jobs in a given workload for constructing an optimized job schedule [16]. For the default method of Hadoop, which cannot schedule the tasks to the nodes with the prefetched data, a prefetching technique was proposed in [17] to hide the remote data access delay caused by the map tasks processed on the nodes without the input data. We proposed MTSD, an extensional MapReduce task scheduling algorithm for deadline constraints in the Hadoop platform [18], which allows a user to specify a job's deadline and tries to make the job to be finished before the deadline. Some of these proposals [19] presented the workload characteristic oriented scheduler, which strives for co-locating tasks of possibly different MapReduce jobs with complementing resource usage characteristics. In [20], the authors presented a scheduling technique for multi-job MapReduce workloads that is able to dynamically build performance models of the executing workloads, and then use these models for scheduling purposes. As the MapReduce distributed computations were analyzed as a divisible load scheduling problem [21], several classes of algorithms were proposed and examined for scheduling divisible loads on a heterogeneous system with memory limits [22]. Some task scheduling algorithms are to release the data communication among remote slots, e.g., the center-of-gravity reduce scheduler is a locality-aware and skew-aware reduce task scheduler for saving MapReduce network traffic [23], and MaRCO employs eager reduce to process partial data from some map tasks while overlapping with other map tasks' communication [6].

Furthermore, there are also many researches using MapReduce to resolve the big data process [24]. A MapReduce-based framework for HPC analytics was developed in [25] to eliminate the multiple scans and also reduce the number of data preprocessing MapReduce programs. Considering the dynamic resource allocation for the IaaS cloud systems, the algorithms in [26] can adjust the resource allocation dynamically based on the updated information of the actual task executions. In these data processes, the utility becomes a considerable problem naturally. The authors of [10] discussed how to increase the utilization of MapReduce clusters to minimize their cost. They optimized the execution of MapReduce jobs on the cluster through the design of a job schedule that minimizes the completion time (makespan) of such a set of MapReduce jobs. To achieve the optimal user utility, the goal of resource provisioning in [27] is to minimize the cost of virtual machines for executing MapReduce applications. For the practical applications, some aspects of reality should be taken into account, such as fault tolerance [28] and energy efficiency [29,30].

In this article, it is not possible to discuss every related scheme. Hence, we only outline a few closely related papers as above. The main focus of most current works about MapReduce scheduling algorithms appears to be job scheduling, less involving task delay, especially the consideration of the tasks with the same key for the input data block. Through analysis of the intermediate data process in Hadoop, this paper indicates that the scheduling of reduce tasks is one of the key problems which affect the performance of a system.

3. Problem analysis

Hadoop allows the user to configure the job, submit it, control its execution, and query the state. Every job consists of independent tasks, and each task needs to have a system slot to run. Fig. 1 shows the time delay and slot resources waste problem in reduce scheduling. Y-axis in Fig. 1 means the resource slots, which is represented by Map slots and Reduce slots. At first in Fig. 1(a), we can know that Job_1 and Job_2 are the current running jobs, before the time t_2, each job is allocated two map slots to run respective tasks. Because the reduce tasks will begin once any map task finishes,