Computers and Electrical Engineering xxx (2013) xxx–xxx

Contents lists available at SciVerse ScienceDirect

Computers and Electrical Engineering

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

Balancing reducer workload for skewed data using sampling-based partitioning ☆

Yujie Xu [a], Wenyu Qu [a,*], Zhiyang Li [a], Zhaobin Liu [a], Changqing Ji [a,b], Yuanyuan Li [a,c], Haifeng Li [a]

[a] School of Information Science and Technology, Dalian Maritime University, Dalian 116026, China
[b] School of Physical Science and Technology, Dalian University, Dalian 116600, China
[c] School of Software, Dalian Jiaotong University, Dalian 116028, China

ARTICLE INFO

Article history:
Available online xxxx

ABSTRACT

MapReduce has emerged as a popular tool for distributed processing of massive data. However, it is not efficient when handling skewed data and it often leads to reducer load imbalance. In this paper, we address the problem of how to efficiently partition intermediate keys to balance the workload of all reducers when processing skewed data. We present a sampling scheme to compute the approximate distribution of key frequency, estimate the overall distribution and then make a partition scheme in advance. Then, we apply it to map phase of the executing MapReduce job. This work not only provides a load-balanced partition strategy, but also keeps a high performance of synchronous mode of MapReduce. We also propose two partition methods based on sampling results: cluster combination and cluster split combination. The experimental results show that our methods achieve a better time and load balancing results.

1. Introduction

With the rapid growth of information and data, there is an urgent need for large-scale data analysis and processing. Especially for many enterprises, they continuously collect large datasets which record valuable information, e.g., customer interactions, product sales, user click stream on the web and other types of information. Then, they make marketing strategies based on many large data analysis activities. The ability to perform scalable and timely analytical processing of these large-scale datasets to extract useful information is now a critical ingredient for success [1].

However, large-scale data analytical processing is a nontrivial work. Fortunately, MapReduce [2], as a popular programming model for parallel processing massive dataset (usually greater than 1TB), has shown its own unique charm in simplicity, reliability and scalability. It has attracted the attention of both industry and academic. Furthermore, it has been extended to many aspects for various purposes [3–6].

MapReduce is a simple programming model that consists of two functions: the map function which transforms input data into ⟨key, value⟩ pairs and the reduce function which is applied to each list of values that correspond to the same key. It is also a parallel processing model that implements "group data by key, then apply the reduce function to each group", i.e., the pairs generated by the map function are grouped by their keys and each group is processed by a single reducer. The group operation is implemented by applying a hash function to each ⟨key, value⟩ pair and assigning a partition number to it.

☆ Reviews processed and recommended for publication to Editor-in-Chief by Guest Editor Dr. Jia Hu.

* Corresponding author. Tel.: +86 041184723505.
E-mail address: wenyu@dlmu.edu.cn (W. Qu).

0045-7906/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.compeleceng.2013.07.001

The default hash function in Hadoop [7], which is the most popular open source implementation of MapReduce, is extremely simple. It is efficient and effective when keys appear equally and distribute uniformly. Using it can make each reducer receive almost the same number of cluster(a set of ⟨key, value⟩ pairs sharing the same key) and the processing time of each reducer is almost the same. For example, assuming there are 3 reducers and 6 clusters, each cluster has 5 ⟨key, value⟩ pairs. If using the uniform hash, each reducer receives 2 clusters and 10 ⟨key, value⟩ pairs. However, this blind hash function fails on the skewed data. Because the size of the cluster is different from each other when processing skewed data. Reducers assigned many larger clusters have a high workload for they need to process more ⟨key, value⟩ pairs and even they become a "straggler". To illustrate this situation, using the above example, the number of clusters is still 6 but the size of each is changed into 1, 8, 5, 10, 80, 15. Reducer1 receives 11 ⟨key, value⟩ pairs, reducer2 receives 88 ⟨key, value⟩ pairs and reducer3 receives 20 ⟨key, value⟩ pairs.

Data skew occurs naturally in many applications. For example, (1) PageRank [8], it is a link analysis algorithm that assigns weights to each vertex in a graph by iteratively aggregating the weights of its inbound neighbors. This application can thus exhibit skew if the graph includes nodes with large degree of incoming edges. (2) Inverted index [9], it is an index data structure sorting a mapping from content, such as words or numbers, to its locations in a database file, or in a document or a set of documents. It also exhibits skew if a content (such as a word) appears in a large number of documents. Other applications, such as data mining, join operation of data, would also involve the skewed data.

With the presence of data skew, the blind hash function is inadequate and brings many problems. First of all, it leads to poor parallelism, unfairness of reducer's input and high difference of reducer's runtime. Secondly, it increases the makespan of a MapReduce job, for a job response time is dominated by the slowest reducer instance and the next job cannot start until all reducers of the previous are completed. Finally, it causes the severe performance degradation. Therefore, there is an urgent need of a load balancing strategy for skewed data.

In order to overcome the above deficiencies and effectively partition intermediate keys to balance the load of all reducers when processing skewed data. We propose a method that divides a MapReduce job into two phases: sampling MapReduce job and executing MapReduce job. The first phase is to sample the input data, compute the approximated distribution of the key frequency and then make a good partition scheme in advance. In the second phase, executing MapReduce job uses this partition scheme in every mapper to group intermediate keys quickly. We design a sampling method to process the input data and we also give a theoretical guarantee of it in this paper. Furthermore, we propose two load-balanced partition schemes according to the sample results: cluster combination and cluster split combination. The idea of the first method is always assigning the cluster with the largest number of ⟨key, value⟩ pairs to the reducer with the smallest number of ⟨key, value⟩ pairs, to achieve the load balancing of all reducers. Considering the situation that the data is seriously skewed, the second method equally splits the larger cluster into n (n is the number of reducer) pieces and then assigns one piece to each reducer. We conduct experimental studies on the above two partition schemes. The results suggest that a significant performance improvement has been achieved by our algorithms.

The rest of this paper is organized as follows. Section 2 introduces the related work. Section 3 is about synchronous map and reduce. We describe the overall design in Section 4. Section 5 presents our sampling method in MapReduce and we also present two partition schemes in Section 6. Section 7 reports experimental results and Section 8 concludes.

2. Related work

Data skew has been well-studied in parallel database, especially the join operation [10,11]. Straggler problem is firstly proposed in [2]. In this paper, Jeffrey Dean and Sanjay Ghemawat proposed using the backup task to alleviate the influence of stragglers. Lin [12] researched the straggler problem in MapReduce and found that it was caused by the Zipf distribution of input or intermediate data. Kwon et al. [8] presented five types of skew that can arise in MapReduce applications and five best practices to mitigate skew in a MapReduce job.

The data skew problem can be solved by making a good partition scheme based on the key's frequency [13–15]. In [13], authors have demonstrated that the presence of partitioning skew causes a huge amount of data transfer during the shuffle phase and leads to a significant unfairness on the reduce input among different data nodes. To overcome these deficiencies, they presented locality-aware and fairness-aware key partitioning scheme in MapReduce. An intermediate key is partitioned based on its frequency and the fairness score calculated after the shuffle phase. And in [14], for the join algorithm in Hadoop does not effectively handle the skewed data, Fariha Atta et al. proposed a skew handling join algorithm. This algorithm replaces the key partitioning with the range partitioning that is capable of handling skew in the input data. YongChul Kwon et al. [15] attempted to minimize computational skew in scientific analysis tasks with black box functions and presented SkewReduce for users to easily express and efficiently execute the complex data analysis. It used an optimizer that determines how best to partition the input data to minimize the impact of computation skew.

An alternative method to handle data skew is assigning keys to reducers based on cost models [16,17]. For the complex MapReduce task on skewed scientific data, Gufler et al. [16] defined a new cost model that took into account non-linear reducer tasks and provided a distributed method to estimate the cost. According to the estimated cost, they also proposed two load balancing approaches: fine partitioning and dynamic fragmentation. For challenges of gathering statics from the mappers, Gufler et al. [17] presented TopCluster. It is a distributed approach for MapReduce-based systems. First, it calculates the local histogram and applies a parameter to determine the head of the local histogram. The final result of TopCluster

Please cite this article in press as: Xu Y et al. Balancing reducer workload for skewed data using sampling-based partitioning. Comput Electr Eng (2013), http://dx.doi.org/10.1016/j.compeleceng.2013.07.001