Simple | Smart | Speed
HOOLS

Research on Computing Efficiency of MapReduce in Big Data Environment

Abstract. The emergence of big data has brought a great impact on traditional computing mode, the distributed computing framework represented by MapReduce has become an important solution to this problem. Based on the big data, this paper deeply studies the principle and framework of MapReduce programming. On the basis of mastering the principle and framework of MapReduce programming, the time consumption of distributed computing framework MapReduce and traditional computing model is compared with concrete programming experiments. The experiment shows that MapReduce has great advantages in large data volume.

1 Introduction

According to the data universe report of International Data Corporation (IDC): in 2008, the global data volume was 0.5ZB and 2010 was 1.2ZB and human beings officially enter the era of ZB. What is even more striking is that the volume of global data will still maintain a high growth rate of 40% every year before 2020. It doubles every two years or so, and is expected to exceed 35ZB by 2020, 70 times more than in 2008 [1]. It can be said that the big bang of data.

Big data, in addition to the huge amount of data, big data also has the characteristics of data diversification, data processing speed, low data value density and authenticity and it is summed up as the "5V" feature of big data [2]. The key technologies of big data processing include acquisition, pre-processing, storage and management, analysis, mining and display, etc. [3]. For big data, traditional computing mode can no longer meet the needs of the new era. Therefore, the new computing mode MapReduce programming framework arises at the historic moment. MapReduce is a model for massive data parallel processing proposed by Google in 2004, which is simple to use, highly scalable and fault-tolerant [4]. Therefore, it is widely used in the development model of parallel computing in large-scale computer cluster. By comparing with the traditional calculation method, the efficiency of MapReduce is analysed, and the application scenario of MapReduce programming framework is given.

2 Related work

MapReduce model is simple, and many problems in reality can be expressed by MapReduce model. As a result, the model has received great attention immediately after its publication, and has been widely used in bioinformatics, text mining and other fields. Lin Yong et al. [5] apply MapReduce technology to the log data analysis of a project platform, and the experimental results prove that MapReduce can solve the inefficiency problem faced by the single machine of Web log processing better, and can integrate computer resources better. Mi Yunlong et al. [6] proposed a parallel algorithm for granular concepts cognitive learning based on MapReduce framework, and established a granular concepts cognitive computing system from the perspective of extension and connotation respectively, and then studied the given object set or attribute set. Zhou Tao et al. [7] proposed a massive image data processing model based on MapReduce parallel framework, and applied MapReduce parallel framework to the field of graphics and image processing. Yang Lingyun et al. [8] proposed a big data financial credit evaluation model based on MapReduce in the supply chain, and applied the MapReduce computing framework to the financial big data domain. In addition, it has been widely studied in many fields, such as information security [9], data audit [10], and text data statistical analysis [5]. Big data processing processes such as app, shopping habits, personality analysis, constellation testing, genetic maps, and social networks are also widely used.

3 MapReduce

The reference model for the overall architecture of big data is shown in Figure 1. The core of the reference model is the Big Data Storage Framework (HDFS) [11, 12] and the Big Data Processing Framework (MapReduce).

[a] **Corresponding author: gtllei@ynufe.edu.cn**

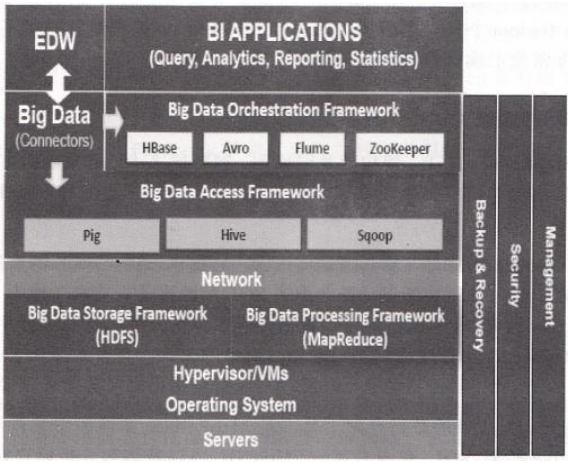

Figure 1. Big data overall architecture reference model

MapReduce is a software framework for distributed parallel computing. It is responsible for the operation of massive data and consists of map function and reduce function, runs on large clusters composed of ordinary PC and is compatible with many languages. Generally speaking, MapReduce is a cheap and general distributed computing framework. Map function is responsible for task decomposition, and reduce function is responsible for the synthesis of the calculated results of the decomposed tasks.

The MapReduce framework runs on key pairs such as key value. Map decomposes the original input into a set of intermediate key-value pairs, the reduce process then synthesizes the result into the final output, which is called a job in MapReduce. A job is composed of several tasks, including a number of map tasks and several reduce tasks. The map task then processes these key-value pairs in complete parallel. The framework sorts the output of the map (that is, the decomposed key-value pairs) and inputs the results to the reduce task.

MapReduce includes not only map and reduce phases. There are also a number of processes between common maps and reduces, including partition, sort, combine, copy, merge, and so on. The specific process is shown in Figure 2. Next, we will introduce the steps of MapReduce by counting the number of words.

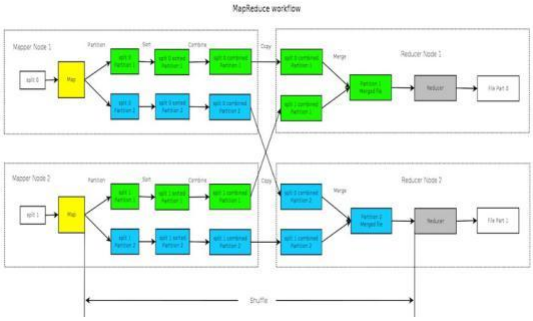

Figure 2. MapReduce workflow

First, create two text files as input examples.

Step 1: map. The two input text files are the two split: split 0 and split 1 in the diagram. The two splits are allocated to two Mapper by default, and each split corresponds to a mapper. In this step, the contents of the file are decomposed directly into words and 1, where the word is the main key, followed by the number 1 which corresponds to the value.

Step 2: partition. According to the difference of Key, separate the data, and ensure the uniqueness of Key. The common Partition method is Hash. In the figure, Partition 1 will be ready for Reducer 1, and Partition 2 is for Reducer 2.

Step 3: sort. In fact, this process is not necessary and can be handled according to the needs of the customers. After partition, not all parts are ordered. To ensure that all parts are ordered, sort step is added.

Step 4: combine. This process occurs after the output of the preceding Map. The aim is to calculate the result before it is sent to Reducer to reduce the size of the file and facilitate subsequent transmission. Also, this step is not necessary.

Step 5: copy. Copy process downloads data belonging to its own partition from the Reducer node to each mapper node via http. This step will prepare data for Reducer.

Step 6: merge. As shown in the previous step, the files that Reducer gets are downloaded from different Mappers, and the merge step merges them into one file.

Step 7: reduce. Based on the above calculation process, the final result is given. It is the last step.

4 Experimental analysis

4.1 Experiment

The content of this experiment is programming to calculate the mean and variance of multiple data. The traditional programming method and MapReduce programming method are respectively calculated. Calculating the amount of data increased from 10 thousand to 10 million, calculating the time consumed by two programming modes. In the experiment, when the initial data was at 10 thousand, MapReduce consumed 0.389 seconds more than the traditional method. With the increase of data volume and the amount of data between 400 thousand and 500 thousand, the two methods consume the same time. As the amount of data continues to grow, MapReduce starts to take significantly less time than traditional methods, and the larger the amount of data, the greater the difference between the two. The calculated data and results are shown in Tables 1 and 3.

Table 1. Calculated data

	1	100	300	400	500	600	700	800	900
T1	0.031	2.781	10.242	12.664	15.995	19.005	21.876	24.320	29.963
T2	0.42	2.234	7.457	8.521	9.664	03.061	15.270	16.181	20.460
Results (T1-T2)	-0.389	0.547	2.767	4.143	6.331	5.944	6.606	8.139	9.563

In the table, the number of data involved in the first operation is 10000. The amount of time consumed in the table is seconds. T1 represents the time consumed by traditional methods, and T2 represents the time consumed by the MapReduce method.

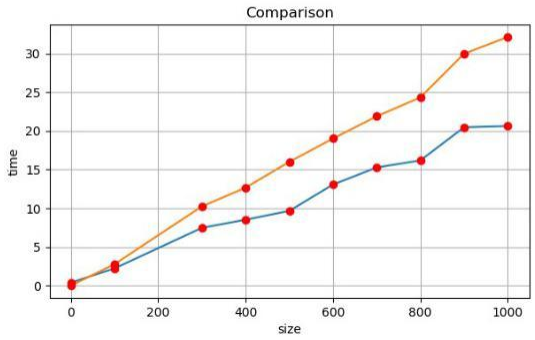

Figure 3. Comparison of calculation results

Figure description:
The magnitude of the abscissa data unit: ten thousand.
The ordinate size unit: second.

4.2 Analysis

In this experiment, we only carry out mathematical operations of mean and variance of data. In the case of small amount of data, the traditional computing method has obvious advantages. But with the increasing amount of data, the difference between the traditional calculation and MapReduce method is shrinking. When the amount of data reaches 400 thousand, the time consumed by the two is the same, and the result is shown in Figure 4. As the amount of data continues to grow, the time consumed between MapReduce and traditional computed queries increases significantly as the amount of data increases, as shown in Figure 5. That is, the larger the amount of data, the more dominant MapReduce computing is, and the less time it takes than traditional computing methods.

The experiments shows that the MapReduce programming mode does not increase efficiency under the condition of small amount of data. Since Map and Reduce processes consume time, they consume even more time. But with the increasing amount of data, the advantages of MapReduce can be reflected. Therefore, to use MapReduce, we need to decide whether to choose according to the actual amount of data.

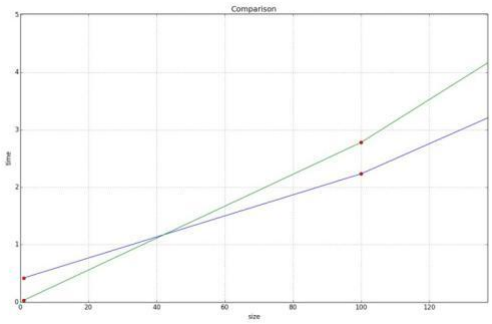

Figure 4. Comparison of calculation results with small amount of data

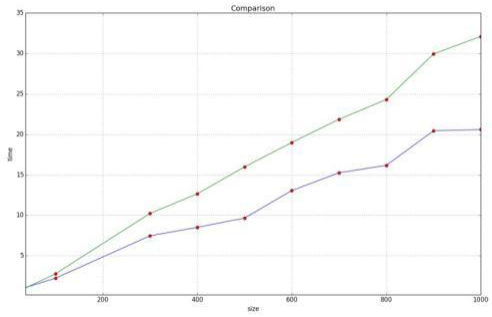

Figure 5. Comparison of calculation results with large amount of data

5 Conclusion

In the face of massive data, the time complexity of traditional computing mode has increased to an unbearable level. Distributed computing, led by cloud computing, has become an important way to solve massive data computing. As the core computing framework of cloud computing, MapReduce becomes the key technology to solve distributed computing. Traditional research has focused on the MapReduce programming pattern itself or the application domain, but little on its efficiency and application environment. For this reason, based on the big data, this paper deeply studies the principle and framework of MapReduce programming. The advantages of MapReduce are verified by concrete experiments, and the advantages are analyzed, and the concrete conclusions are given: for the actual operation, in the case of small amount of data, because Map, Reduce, and distributed machine-to-machine communication processes themselves take a certain amount of time, the total computational time will exceed the traditional computational method. When the amount of data is large enough, MapReduce can show its huge advantage in computing power. The larger the amount of data, the more obvious the advantage is. However, this paper only verifies the advantage of MapReduce by calculating the mean and variance of massive data, and there is still a certain distance from the actual application scenario. Subsequently, we will transfer the operation to the actual application scenarios to study how MapReduce can better play its computing advantages in practical applications.

Acknowledgement

This work was supported by National Natural Science Foundation of China (Nos.61379032, 61763048, 61263022, 61303234, 61662085), National Social Science Foundation of China (No.12XTQ012), Science and Technology Foundation of Yunnan Province (No.2017FB095), Yunnan Province Applied Basic Research Project(No.2016FD060), Science Research Project of Yunnan Education (Nos.2017ZZX001, 2017ZZX227), Key Project of Scientific Research of Yunnan Education (2015Z018), Provincial Scientific and Technological Innovation Team Project of Yunnan

University (2017HC012), the 18th Yunnan Young and Middle-aged Academic and Technical Leaders Reserve Personnel Training Program (No.2015HB038), Research Project of Yunnan University of Finance and Economics (N0. 80025092472).

The authors would like to thank the anonymous reviewers and the editors for their suggestions.