Optimizing Performance of Hadoop with Parameter Tuning

Xiang CHEN[1,a], Yi LIANG[1,b], Guang-Rui LI[2,c], Cheng CHEN[1,d], and Si-Yu LIU[1,e]

[1]*Faculty of Information Technology, Beijing University of Technology, Beijing, China*
[2]*NO.10 Building, Suzhou Software Park, No.78, China Mobile Software Technology Co., Ltd, Suzhou, China*
[a]*chenxiang1@emails.bjut.edu.cn,* [b]*yliang@bjut.edu.cn,* [c]*liguangrui@cmss.chinamobile.com,* [d]*ccheng@emails.bjut.edu.cn,* [e]*liusifen@emails.bjut.edu.cn*

Abstract: Optimizing Hadoop with the parameter tuning is an effective way to greatly improve the performance, but it usually costs too much time to identify the optimal parameters configuration because there are many parameters. Users are always blindly adjust too many parameters and are sometimes confused about which one could be changed at a higher-priority. To make optimization easier, classifying the parameter based on different applications will be helpful. In this paper, we will introduce a method that can classify these parameters in order that users can optimize performance more quickly and effectively for different applications.

1 Introduction

With the rapid development of the Internet, the era of big data has arrived as the amount of data on the web has been expanding dramatically. MapReduce was firstly proposed by Google in 2004 and has been a popular parallel computing model for big data processing [1]. The core ideal of MapReduce platform is to store and compute mass data based on distributed computational nodes, and to realize localization process of data through task scheduling so as to improve the efficiency of data process. The most popular implementation of the MapReduce model is the Hadoop framework [2]. Hadoop is an open source software platform managed by the Apache Software Foundation and has proven to be very helpful in storing and managing vast amounts of data cheaply and efficiently. Users can develop and run applications easily on systems with thousands of nodes by Hadoop. Now Hadoop is widely used in industry and commerce for many big data related tasks.

As Hadoop is widely used to process big data offline [2][3], improving the Hadoop's performance is meaningful and important both in academic area and industry. The excellent performance means effectiveness and hign throughput of the Hadoop.

The performance of the application is closely related to its configuration on Hadoop. There is a large number of parameters in the configuration and they need the user to set them. A bad configuration of the inappropriate parameters' values will seriously affect the performance. It will take much time for users to change the parameter's value to achieve a better performance. What's more, users always tune these parameters with lack of guidance and may result in the poor performance. To execute a different application, they may need to repeat this procedure.

From the above, in this paper we propose a solution for this problem. We introduce a method that can classify these parameters into three categories, the parameters in the same group can significantly impact on one category of application. User can tune these parameters firstly to reduce the main bottleneck of the application. It can be much more effective and save much time.

2 Significance Parameters

Parameter Name	Brief Description	Range of Value
mapred.map.tasks	Number of map tasks	[6,24]
mapred.reduce.tasks	Number of reduce tasks	[6,24]
mapred.compress.map.output	Compress the maps' outputs or not	[true,false]
mapred.child.java.opts	Java opts for task tracker child processes	[256MB,2048MB]
io.sort.factor	Number of sorted streams to merge during sorting	[10,50]
io.sort.mb	Buffer memory to use while sorting files	[128MB,1024MB]
dfs.block.size	Data size processed per mapper task	[32MB,256MB]
dfs.replication	Block replication number	[1,3]

Hadoop parameters control various aspects of job behavior during execution. About 200 parameters can be set by users, but not all the parameters have great impact on the Hadoop, and some of them even have no impact on the performance. As a conservative estimate, the settings of more than 25 of these Hadoop parameters can have significant impact on application performance [4]. To reduce the difficulty of our experiments, we chose some of them as examples and show them in Table 1.

The parameter with appropriate value will bring high performance for Hadoop. We set the range of each parameter because too large or too small of the parmeter values will obviously affect the perforamnce. User can optimizing performance of Hadoop with parameter tuning in the range. But user still should select the useful paramters for different application to improve the performance, the guidance for parameter selection could be valuable.

In the following of this paper, we will describe the classification method in detail, then we evaluate the impact of our method by experiments.

3 Classification Method

In this section we will describe our classification method. To make our method suitable for most case, with different application, we have different conclusions of optimization. For a given application, we usually can divide it into three categories by its bottleneck, that is IO-bound, CPU-bound and hybrid [5]. Every category of application has a corresponding parameters class for optimizing. Based on it, these parameters are generally divided into four categories: IO-tuning, CPU-tuning, both-tuning and none. It is obvious that IO-tuning parameters are used for IO-bound application, CPU-tuning parameters are used for CPU-bound application and both of them are used for hybrid application.

We can run a typical application to help us evaluate the impact of the parameter. In our experiment, we focused on the metrics of execution time and it can reflect the performance. Then our classification method shows the following: First of all, we chose a typical IO-bound application, such as Sort. To evaluate a parameter, we changed the parameter's value in the range while the other parameters kept unchanged. Then we executed the same application several times with different value of this parameter and got some experimental data. To cut errors, we can repeat this experiment more than once and take the average value for further analysis. We defined a new metrics δ to denote the importance of this parameter on the application, it is calculated by this formula:

$$\delta = \sqrt{\frac{\sum_{i=1}^{N}(T_i - T)^2}{N}} \quad (1)$$

T is the average execution time of the experiment, Ti represents that the *ith* Execution time for this parameter, N is the times of experiments. We can set a threshold value for the δ, if the δ greater than the threshold, we think this parameter have a great impact on this application, it is worth optimizing this category of application by this parameter. The application is IO-bound, and so this parameter will be classified to IO-tuning parameters.

The threshold value is set by user, with different decision of threshold value may result in the difference of the classification result of parameters. The inappropriate values of threshold may affect the optimization effect. More experimental result will increase the accurate of threshold selection.

In this way, we can classify all the parameters into four categories.

4 Experiment and Evaluation

4.1 Environment

For simplicity, we ran experiments with Hadoop cluster configurations of one master and one slave. The experiments were performed under Hadoop 1.2.1, the new Hadoop version was not used in the experiment because the resource management framework YARN in Hadoop 2 provided a unified resource management and scheduling for the application. The configuration space compared to the 1.2.1 version is less, Using the lower version Hadoop to facilitate the show the optimization of effect.

Each machine has two 6-core 2.4 GHz Intel Xeon E5645 processors and 16GB of memory. They all ran Ubuntu 14.04.3 LTS with kernel 14.04 operating system and ext4 file system. For the experiments in this paper, to make the application runtime short, the input data of all experiments was about 16GB.

We ran our experiment with typical IO-bound Sort benchmark and CPU-bound Kmeans benchmark to help us classify these predefined parameters. Then we tried to optimize the CPU-bound Wordcount benchmark by the classification. All the workload is generate by the BigDatabench [6].

4.2 Classify These Parameters

Table 2~4 shows the experimental data of running Sort benchmark while changing the value of specified parameters and keep the others unchanged. Due to limited space, we didn't list the experimental result of all the parameters and have only shown the typical parameter which is easy to estimate.

Table 2. Execution time of change the value of mapred.map.tasks and keep the others unchanged

mapred.map.tasks	4	6	8	10	12
execution time (s)	1346.43	1312.16	1444.8	1277.86	1294.43

Table 3. Execution time of change the value of io.sort.mb and keep the others unchanged

io.sort.mb (MB)	128	256	512	1024	2048
execution time (s)	1223.9	1242.0	1294.2	1261.2	1100.8

Table 4. Execution time of change the value of dfs.block.size and keep the others unchanged

dfs.block.size (MB)	32	64	128	256	512
execution time (s)	1196.1	1095.6	1057.3	1145.9	1230.5

We changed the parameter's value in the range while the other parameters kept unchanged. Then the execution time of Sort benchmark is changed by this parameter tuing and the shorten time means the performance gains of our optimization work. The Sort benchmark is IO-bound, the performance bottleneck of Sort is the imited bandwidth of disk. The parameter which can improve the disk bandwidth or the processing effeciency of Hadoop will significant improve the performance of this type of application.

In this way, with the experimental result, the δ of these parameters can be obtained. The δ of these parameters respectively is 59.36188, 66.05308 and 63.32071. The difference between them is not large and threshold value is set to 30, all of these parameters are large than the threshold value and they all be regarded as IO-tuning parameters.

Table 5~7 shows the experimental data of running Kmeans while changing the value of specified parameters and keep the others constant.

Table 5. Execution time of change the value of mapred.map.tasks and keep the others unchanged

mapred.map.tasks	8	10	12	14	16
execution time (s)	658.31	634.28	633.54	627.46	653.38

Table 6. Execution time of change the value of io.sort.mb and keep the others unchanged

io.sort.mb (MB)	128	256	512	1024	2048
execution time (s)	622.36	613.27	606.85	614.89	610.91

Table 7. Execution time of change the value of dfs.block.size and keep the others unchanged

dfs.block.size (MB)	32	64	128	256	512
execution time (s)	884.59	614.89	685.08	476.24	454.16

The Kmeans benchmark is CPU-bound and the performance bottleneck of Sort is the low efficiency of processing data. Only the parameter which can improve the processing effeciency of Hadoop can significant improve the performance of this type of application. And this can guide the decision of CPU-tuning parameter.

From experimental results, we can obtain the δ of these parameters. The δ of these parameters respectively is 12.13479, 5.12432 and 156.45732. Because Kmeans is CPU-bound, it is easy to conclude that mapred.map.tasks and io.sort.mb is not CPU-tuning parameter, and dfs.block.size is not CPU-tuning parameter.

In summary, mapred.map.tasks and io.sort.mb are CPU-tuning parameters, dfs.block.size is both-tuning parameter. With the experiments, all the parameters can be classified to corresponding categories.

4.3 Evaluation

We tried to optimize Naïve Bayes benchmark by the classification result, and the experimental results shown as Figure 1. In the X-axle, the number means tuning one paramter in the range while the others constant. The distance between height of point A and height of point B is the execution time reducing by tuning this parameter, so the downtrend means the optimization effect of the parameter.

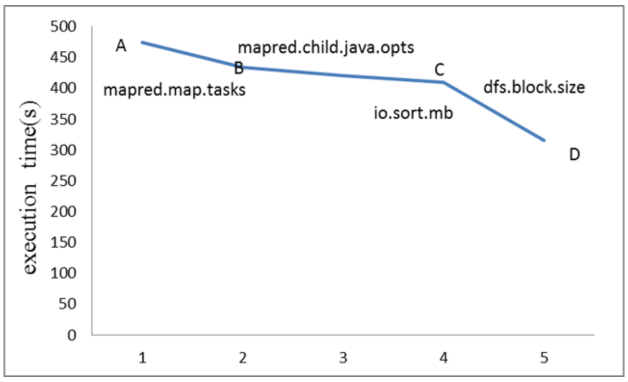

Figure 1. impact of paramters on Naïve bayes

In Figure 1, for the CPU-bound application, the useful parameters should be the CPU-tuning or both-tuning parameters, such as dfs.block.size. The downtrend betwent C and D is the execution time reducing by the dfs.block.size tuning. It is about 23.2% shorted than before and it is the maximum value in our optimization work. It means optimizing Hadoop with dfs.block.size tuning can be a most effective way to improve application's performance, it demonstrates the helpful of our classification method. The classification result provides the optimization guidance for users.

Summary

In this paper, we focuses on the performance of Hadoop big data processing platform and introduce a classification method for different types of applications. It can help user optimize Hadoop application with the parameter tuning in a quickly and effective way.

Acknowledgment

I would like to express my gratitude and appreciation to Professor Yi Liang for great efforts she had put into this paper. Her enlightening advice and continued support have made this research work a valuable experience for me.

References

1. Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters[C]// Conference on Symposium on Opearting Systems Design & Implementation. USENIX Association, 2004:10-10.
2. Apache Hadoop [P/OL]. Available: http://hadoop.apache.org/.
3. White T, Cutting D. Hadoop : the definitive guide[J]. O'reilly Media Inc Gravenstein Highway North, 2009, 215(11):1 - 4.
4. Babu S. Towards automatic optimization of MapReduce programs[C]// Proceedings of the 1st ACM Symposium on Cloud Computing, SoCC 2010, Indianapolis, Indiana, USA, June 10-11, 2010. 2010:137-142.
5. Wang L, Zhan J, Jia Z, et al. Characterization and Architectural Implications of Big Data Workloads[J]. 2015:145-146.\bibitem{RefJ}
6. Wang L, Zhang S, Zheng C, et al. BigDataBench: A big data benchmark suite from internet services[C]. International Symposium on High PERFORMANCE Computer Architecture, 2014:488-499.