ATLANTIS PRESS

Advances in Computer Science Research, (ACSR), volume 52

2016 International Conference on Computer Engineering and Information Systems (CEIS-16)

Improving Performance in Hadoop Using Automatic and Predictive Configuration

Juan Fang, Hao Sun, Li-Fu Zhou, Xing-Tian Ren, Min Cai

College of Computer Science,

Beijing University of Technology,

Beijing, China

E-mail: fangjuan@bjut.edu.cn, 15810484755@yeah.net, cliffneo009@gmail.com, renxt, min.cai.china@bjut.edu.cn

***Abstract*-MapReduce is an effective programming model for analyzing large-scale data. Hadoop-a distributed processing system is widely used nowadays. Improving the task parallelism can be a key point to improve the MapReduce performance in Hadoop. In this paper, we address the problem in two ways. On the one hand we can run the tasks with some dynamic configurations. On the other hand, considering of the difference of tasktracker we use mathematics method to predict the cups' utilization of tasktracker to assign the task. Experimental results on both ways show we can improve the performance in Hadoop by improving the task parallelism.**

Keywords- high performance; dynamic; prediction; big data

I. INTRODUCTION

The MapReduce framework[1] provides a transparent programming model for us to analyse large-scale datasets. It can be easy for users to learn and use. The data scale is becoming larger than ever before. On the web we produce Petabyte-scale everyday [2]. We can dig for information from the datasets. On other fields like science analysing large-scale data can also help a lot to come into being scientific discovery. How to improve the performance in MapReduce can have a great value in our future life. The MapReduce can also be available on cloud computing platforms.

Hadoop[3] uses a distributed file system to store data which is going to be computed. The distributed file system called HDFS is fault-tolerant. It makes copies of blocks in the distributed disks called datanode. When we run a MapReduce job, the tasktracker will assign the task towards the data stored in the datanode. In this way we reduce the data transmission among datanodes. To make the task assigned to the tasktracker correctly can be the key point to improving the performance in MapReduce. Hadoop is widely used in web indexing, data mining, scientific simulation and so on.

To run a job in the MapReduce framework in Hadoop, we must set the map and reduce classes in the program. The map class is used for map tasks and produces output for reduce tasks. The reduce class will get the output of the map tasks and analyze the results of the job. Before running a job, Hadoop provides a lot of parameters for us to config such as the maximum number of map tasks running in the tasktracker, block size which is going to be used in producing splits for the map tasks, io.sort.mb which is configuring the space in the memory as the buffer area for map tasks. Hadoop provides more than a hundred parameters for users to config. In order to improve the performance in Hadoop, we cannot ignore these parameters. As we can see some parameters are relevant. We can adjust the relevant parameters in the program.

Hadoop runs the jobtracker to assign the tasks. The job is going to be split into a lot of tasks. The jobtracker collects the information form the tasktracker in Hadoop which is actually running the tasks to assign the tasks. Hadoop uses a communication mode called heartbeat to send information about tasktracker to jobtracker. When the tasktracker has free slot, it sends a heartbeat to jobtracker. The heartbeat contains the hostname, IP address, the name of the tasktracker and other status information about the tasktracker. The jobtracker checks out the information to decide whether to assign the task to tasktracker or not. In this paper, the cups' utilization of the tasktracker is taking into considering about assigning the tasks. Using mathematics method to predict the utilization of cpu of the tasktracker and add the predictive value to the heartbeat. In both ways, Experiment results show we can improve the performance of MapReduce in Hadoop.

II. RELATED WORK

The MapReduce framework in Hadoop is widely used in many fields. As the society is growing faster and faster, Concepts about environmental protection is becoming more and more important. It turns out Green Hadoop [4]. Green Hadoop tries to complete the processing of data meanwhile using the minimum of energy. It presents a detailed evaluation and sensitivity analysis of an energy-conserving, highly scalable variant of the Hadoop Distributed File System(HDFS) called GreenHDFS. GreenHDFS logically divides the servers in a Hadoop clusters into HOT and Cold zones. It shows that GreenHDFS can definitely reduce the energy cost.

Hadoop provides more than a hundred parameters for us to config. Users often run into performance problems because they do not even know that these parameters exist [5]. We can improve the performance of tasks by setting theses parameters. It puts forward a new solution to optimize Hadoop-changing the configuration dynamically[5]. It brings out a new conception for us to improve our performance in program. Among these parameters some parameters are relevant which means setting these parameters together will make better performance of our task. It studies the main factors which influence the performance of Hadoop [6]. It gives the solution of how to solve the problem. MapReduce can achieve better performance with the allocation of more compute nodes from the cloud to speed up computation [6]. A good data placement policy can also improve the

performance of Hadoop. It turns out a data placement policy to store the datasets to improve the performance of Hadoop[7]. The current Hadoop implementation assumes that computing nodes in a cluster are homogeneous in nature. It takes the ability of computing into considering to assign the datasets. It shows ignoring the data-locality issue in heterogeneous can definitely reduce the performance of MapReduce.

III. IMPROVE THE PERFORMANCE IN HADOOP

In this section, we study the relevant parameters provided for us and how to change these parameters in Hadoop dynamically to improve the performance of our tasks in section A. We take the cups' utilization into consideration when assgining the tasks to make the job running parallel in section B.

A. Config the Hadoop Dynamically

Hadoop provides more than a hundred parameters for us to config. How to config these parameters correctlly will definitely improve the performance of Hadoop. Hadoop runs namenode to manage the files stored in the distributed file system called HDFS.We can consider the namenode as a kind of intermediator between job client and datanodes. Namenode splits the file that we want to store in HDFS into blocks. It connects the datanode which is going to store the blocks and gives the client permission to the certain datanode. The client gets the permission of the datanode so that it can pass the blocks to datanodes. The namenode takes a FSimage to log where these blocks stored. When the namenode starts up, it loads the FSimage so that we can find the file we want in HDFS. There comes a problem. When we produce a lot of blocks to store in HDFS, The FSimage stored in namenode is too large. When the namenode loads the FSimage into memory, it will cost a lot of space. If we get 10000000 blocks to store in HDFS, it requires 2GB space in namenode. If we get one hundred million blocks to store, it requires 20GB space in namenode. It can also be very difficult to location the blocks on the condition the FSimage is too big. There comes to the parameter-blocksize which the namdenode uses to split the file that we want to store in HDFS. The job client submits a job and reads the configuration files to get the block size configured. It runs a method to split the job. We can dynamically config the block size to make the performance of Hadoop better. We give the block size a default-value. If we are going to run a large-scale data job, we dynamically change the block size larger.

We do the exact same job with 5GB datasets and 1GB datasets. Results shows that it does improve the performance of Hadoop when we run a large-scale datasets job with a larger block size.

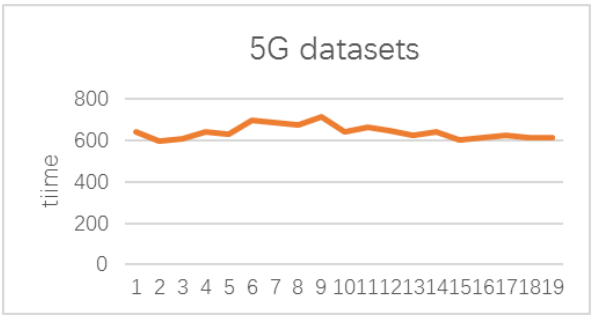

Figure 1. Performance of 5GB datasets

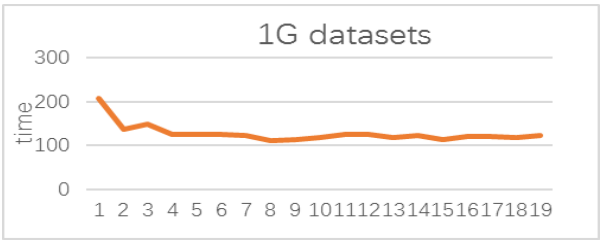

Figure 2. Mance of 1GB datasets

The clusters in Hadoop are becoming larger and larger. Hadoop assumes that the computing nodes are exact the same with the same ability to compute. Actually when the clusters become larger, the old machines do not get the same ability comparing to the new machines. If we do not take this factor into consideration, we cannot make high performance of Hadoop. So when we config the cluster, we must config different in different machines. Hadoop provide a parameter for us to config the maximum number of map tasks running in a traktracker. If we config the tasktracker with different computing ability with exactly the same value, we cannot make full use of the ability of the tasktracker. We can store a configuration file in the clusters. When we start up a job, we read the file into the distributed file system cache so that every machine can read the file. We set every machine with a basic value. If a machine has a better ability like memory, we dynamically set the maximum number of map tasks larger. In this way, we can make full use of computing ability of the clusters to improve the performance of MapReduce in Hadoop.

B. Predict the Cups' Utilization to Assign the Task

The current Hadoop implementation assumes that computing nodes in a cluster are homogeneous in nature. We can definitely improve the performance of MapReduce in Hadoop only if we take the fact that the clusters are heterogeneous. The tasktracker communicate with jobtracker through heartbeat. The tasktracker first collects its own status information like free disk space, host name, free slot. It sends this information to the jobtracker and the jobtracker checks out this information to decide whether to assign the task to the tasktracker or not. Hadoop provides a parameter to configure for us to set an interval to send the information between the jobtracker and the tasktracker. We take the utilization of cups into consideration while sending the

information between the jobtracker and the tasktracker. If the jobtracker checks out the information of certain tasktracker and finds that it got a high cups' utilization. The jobtracker does not assign the task to the certain because the tasktracker is overloaded. If the jobtracker finds the certain tasktracker got a low cups' utilization, it will assign the task to the tasktracker. We collect the cups' utilization before this heartbeat. Using a mathematics method to predict the cups' utilization to send to the jobtracker. The jobtracker gets the predictive value of cups' utilization to decide whether to assign the task or not.

The mathematics is called Least squares. It is a king of optimization of math. It is a standard approach in regression analysis to the approximate solution of overdetermined systems, i.e., sets of equations in which there are more equations than unknowns[8].It is a kind of solution minimizes the sum of the errors made in the results of every single equation. It has a great value in data fitting. The best fit in the least-squares sense minimizes the sum of squared residuals, a residual being the difference between an observed value and the fitted value provided by a model. We first read the log file to collect the cups' utilization to get several points. The data set consists of n points (data pairs) (X_i, Y_i), i = 1, …, n, where X_i is an independent variable and Y_i is an independent variable whose value is read from the log file. The model function can be like this- F (x, y). The goal is to find the parameter values for the model best fits the data when its squared residuals is a minimum.

$$S = \sum_{i=1}^{n} r_i^2 \tag{1}$$

If we get m points for the log file. We get the sample size – M, w_i – the weight of every point occupied proportion in sample size. We get the equation set in our program like this below.

$$\begin{Bmatrix} \varphi_{11}a + \varphi_{12}b = f1 \\ \varphi_{21}a + \varphi_{22}b = f2 \end{Bmatrix} \tag{2}$$

We can calculate the parameters in the equation below.

$$\varphi_{11} = \sum_{i=1}^{n} w_i \tag{3}$$

$$\varphi_{12} = \sum_{i=1}^{n} w_i x_i \tag{4}$$

$$\varphi_{21} = \varphi_{12} \tag{5}$$

$$\varphi_{22} = \sum_{i=1}^{n} w_i x_i^2 \tag{6}$$

$$f1 = \sum_{i=1}^{n} w_i y_i \tag{7}$$

$$f2 = \sum_{i=1}^{n} w_i y_i x_i \tag{8}$$

Every time the tasktracker has free slot, which means its running number of tasks is less than its maximum tasks we configure before. We calculate the cups' utilization this time and write it into the log file right now.

After then we read the log file and create a number list with the cups' utilization. We send the list to a method which will turn out the predictive cups' utilization next period. The method works exactly like the equation sets we talk about above. When we get the predictive value, we send it to the jobtracker. If the jobtracker checks out the certain tasktracker is overloaded, it does not assign the task to the tasktracker. If the tasktracker is overloaded, it will not get high performance in computing and other tasktracker must wait the slowest tasktracker to finish the task. If we do not make the tasktracker work overloaded, we can finish the job parallelism and definitely improve the performance of the job.

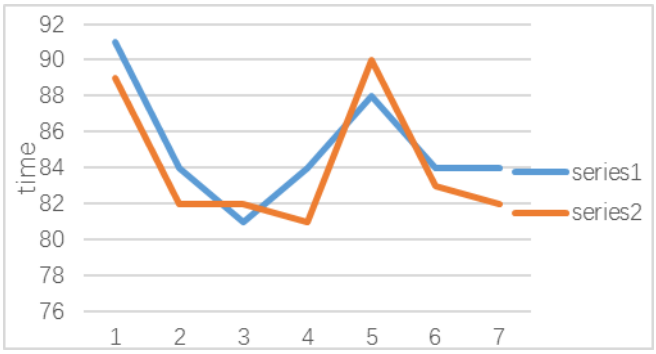

Figure 3. Performance of datasets.

In the figure above, we run the same job. The series 1 shows the results that we do not take the cups' utilization into consideration, the series 2 shows the results that we take cups' utilization into consideration when assigning tasks.

IV. CONCLUSIONS AND FUTURE WORK

The MapReduce framework is widely used in analyzing large-scale datasets. In this paper, we study the configuration parameter which will change the performance of computing in Hadoop. We can configure some parameters dynamically to get high performance of computing. We take the cups' utilization into consideration when assigning the tasks to make the job finish parallel. The MapReduce framework can also be available on cloud computing platforms. In the future, the web application will be built on many kinds of cloud platforms. So how to improve the performance of large-scale datasets computation can have a great value in our future life. We

will study how to improve the performance of data processing in cloud platforms in the future work.

ACKNOWLEDGMENT

This research was supported by the Beijing Municipal Science and Technology Project (Grant No. Z151100002615032)

REFERENCES

[1] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In Proc. of OSDI, 2004.

[2] A. Thusoo et al. Hive - A Warehousing Solution Over a Map-Reduce Framework. PVLDB, 2(2):1626–1629, 2009.

[3] Apache Hadoop. http://hadoop.apache.org/.

[4] Rini T. Kaushik, Milind A. Bhandarkar, Klara Nahrstedt. Evaluation and Analysis of GreenHDFS: A Self-Adaptive, Energy-Conserving Variant of the Hadoop Distributed File System. In Proceedings of CloudCom'2010. pp.274~287

[5] Shivnath Babu: Towards automatic optimization of MapReduce programs. SoCC 2010: 137-142

[6] D. Jiang, B. C. Ooi, L. Shi, S. Wu: The Performance of MapReduce: An In-depth Study. Int'l Conference onVery Large Data Bases (VLDB), 2010

[7] Jiong Xie, Shu Yin, Xiaojun Ruan, Zhiyang Ding, Yun Tian, James Majors, Adam Manzanares, and Xiao Qin:Improving MapReduce Performance through Data Placement in Heterogeneous Hadoop Clusters Department of Computer Science and Software Engineering

[8] https://en.wikipedia.org/wiki/Least_squares

[9] [9] Data pipeline in MapReduce Jiaan Zeng and Beth Plale School of Informatics and Computing

[10] [10] M. Zaharia, A. Konwinski, A. Joseph, R. Katz, and I. Stoica. Improving mapreduce performance in heterogeneous environments. In OSDI, pages 29–42,2008.

[11] [11] A. Pavlo, E. Paulson, A. Rasin, D. J. Abadi, D. J.DeWitt, S. Madden, and M. Stonebraker. Acomparison of approaches to large-scale data analysis. In SIGMOD, pages 165–178. ACM, 2009.

[12] [12] CoHadoop: Flexible Data Placement and Its Exploitation in Hadoop
Mohamed Y. Eltabakh∗, Yuanyuan Tian∗, Fatma Ozcan ¨ ∗Rainer Gemulla+, Aljoscha Krettek#, John McPherson∗