ISBN 978-981-11-0008-6
Proceedings of 2016 6th International Workshop on Computer Science and Engineering (WCSE 2016)
Tokyo, Japan, 17-19 June, 2016, pp. 738-742
doi: 10.18178/wcse.2016.06.132

Hadoop Local Tasks Scheduling Optimization Algorithm Based on Logistic Regression Model

Shuai Renjun[1], Shen Yang[1+], Chen Ping[2], Pan Jing[1], Dong Yanan[1]

[1] School of Computer Science and Technology, Nanjing Technology University, Nanjing 211816, China
2 Nanjing Health Information Center, Nanjing 210003, China

Abstract. For a TaskTracker has multiple local tasks available, by default, the scheduler executes those tasks in succession with the order of the tasks to be found, this is inefficient. In order to optimize the local tasks scheduling, this paper presented Hadoop local tasks scheduling optimization algorithm based on Logistic regression model. First, related feature vectors of the local tasks were selected and defined, then, based on the way of machine learning with Logistic regression model, trained these vector to get the weight of each vector to decide the task priority, and updated the model constantly by the overload rules. The experimental results show that the proposed algorithm improves map task data locality, at the same time of reducing job running time.

Keywords: Hadoop, MapReduce, local tasks scheduling, task priority , overload rules, Logistic regression model three.

1. Introduction

As a kind of open source cloud computing platform, the Hadoop has been paid much attention in recently years. The Hadoop developed the GFS and the MapReduce of Google to HDFS (Hadoop Distributed File System) and Hadoop MapRedece, respectively. It should be noted that the Haddoop MapReduce is a new type of parallel computing model, which has been applied in the core part of computing platform for its advantages of efficiency, simpleness and etc. As an important part of MapReduce, the scheduling algorithm obviously has been paid wide attention for its performance could directly affect the Mapeduce. The network broadband is scarce in the Hadoop platform, thus improving the locality task seems much important. There are several related jobs should be introduced. In [1], Fisher et al. proposed tasks scheduling algorithm based on stream; In [2], Jin et al. introduced a novel method based on Bar ; Zahar et al. [3] proposed delay scheduling algorithm; Zhang et al. [4] and Gu et al. [5] respectively presented the Next-K-Node and Three-Queue scheduling algorithm. However, all these mentioned jobs does not take the characteristics of the local task into consideration and ignore the distribution in data processing, thus we could not sufficiently employ the benefits comes from better tasks in performable local task list.

In [6], Suresh et al. proposed An Optimal Task Selection Scheme (AOTSS). The AOTSS selects better tasks for scheduling based on the sum of feature weights, which can effectively improve the processing of task localization. However, the methods of feature extraction is too simple to most complex applications of big data, especially when we calculate the feature weights. As the feature weights need complex test according to empirical calculation, the final results may not be accurate enough. Inspired by these jobs above, we introduce Logistic regression model to implement the priority ranking for local tasks, which can powerfully select and define the feature vectors related to the tasks. Then, we can obtain the weights for all vectors based on machine learning of Logistic regression model and further implement the priority ranking.

[+] Corresponding author. Tel.: + 86 15195999075.
E-mail address: sy1234@njtech.edu.cn.

Finally, the model would keep updating via the rule of overload, which can both improve the data locality of Map tasks and decrease the job time.

2. Hadoop MapReduce

Hadoop MapReduce is a subordinative construction model composed of a major node JobTracker and several slave nodes TaskTracker. The JobTracker is responsible to divide the jobs provided by clients into several tasks and these tasks will further be regulated. Each TaskTracker includes several Map slots and Reduce slots, the slot can execute the corresponding Map task or the Reduce task. Based on the Heart Beat rule, when there exists spare slot in the process of job, the TaskTracker will communicate with the JobTracker and request for tasks. After the execution has finished, the TaskTracker will also communicate with the JobTracker and report the execution results. It should be noted that the JobTracker always passively accept the information from TaskTracker. Meanwhile, the task executed by the spare node of TaskTracker is decided by the TaskScheduler in JobTracker. The task locality execution means that the JobTracker should divided jobs into some Map task for the execution to the stored nodes of task data, which is the key point of TaskScheduler. As all the Reduce tasks need to read the output of all the Map tasks, we schedule the Reduce tasks to the task nodes in the cluster, and the Internet cost does not change relatively. Therefore, this paper is mainly focus on the scheduling mechanism.

3. Algorithm Design

When there exist spare slots, the TaskTracker will send HeartBeat to request for jobs. After the jobs have been assigned, the Hadoop TaskScheduler will firstly search the locality tasks from the unassigned task list in the jobs. If the locality tasks can be found, TaskScheduler would immediately load and operate these tasks, otherwise, the rest tasks would be operated. Generally speaking, the first locality task loaded by Hadoop is the first locality task that be found. For a TaskTracker, assuming that there are several locality tasks, the throughput rate would decrease, which may affect the operational efficiency to the whole system. For an application of big data, the locality tasks are huge in quantitative terms, thus we can obtain much benefits from the improved scheduling problems.

In order to select the "master" tasks, we need to obtain related features that can affect the priority. Meanwhile, these learned features can have different effects to the priority, thus for the measure to the priority, we need to preset the weights to these features for the locality tasks. The key point to the measure of priority is based on the overload.

Thus there are three important parts for the whole algorithm: the selection of feature vectors, the allocation of overload rule and the acquisition of weights.

3.1. Selection and definition to feature vectors

Definition 1) Replica-Num

Hadoop can divide the input file into several blocks based on HDFS. After this step, each block will be copied by Hadoop. The HDFS will decide to produce or delete the block based to the replication algorithm according to the overload situation, which leads the number of replication for each block can be different. Compared with the block which has more replication, the tasks to the block which has fewer replication can be much more urgent. In addition, after the JobTracker has arranged tasks for Map, the node will cope the block requested by this tasks into the locality storage according the principle of proximity, thus the number of block can much affect the chance of locality, and we need preferential take the block which has fewer copies into consideration. This paper has further defined the copied weights Replica-Weightage as follow:

$$RW = \frac{RN - \min(RN)}{\max(RN) - \min(RN)} \tag{1}$$

where the min (RN) and the max (RN) respectively represent the minimum number and maximum number of replication. The RN can be obtained from the system, and the value is between 0 to 3. Similarly, the value of RW is between 0 to1. This paper cast the RW as the training input of classifier.

Definition 2) Task-Waittime

About the jobing mechanism to Hadoop MapReduce, there are two situations for the assigned task to TaskTracker. One is when all the TaskTracker in the preliminary round can have the spare slots, it can preferential ask for the locality tasks via Pull communication model, which belongs to the startup phase. The other is when TaskTracker does not have the spare slots, we need to estimate the Task-Waittime, which means that we need preferential take the TaskTracker that has the longest Task-Waittime into consideration, thus the TaskTracker can be released quickly. Moreover, we can improve the hit rate of behindhand tasks and control the number of replication tasks, thus we can further reduce the response time, reject the shaking and improve the locality for the tasks. Assuming we have the implementation schedule progeress, the current system time currenttime, the scheduling time scheduletime, and the progressing rate progressrate, we can calculate the Task-Waittime as follows:

$$TW = \frac{1 - progress}{progressrate} \tag{2}$$

$$progressrate = \frac{progress}{(currenttime - dispatchtime)} \tag{3}$$

Definition 3) Disk-Load

As the replications of block are kept in different disks and different nodes, if the data of uploaded task is reserved in the minimum disk load of locality node, the pretty high conversion rate can be realized in terms of disks, which means that we can obtain much better locality tasks. We have the following calculation:

For a node, the load on the disk equals the sum of request to IO and the internet IO cost of non-locality tasks. For simplification, we set the number of slots in the node as Ns, the number of running tasks NR, thus the Disk-Load (DL) can be obtained:

$$DL = 1 - \frac{NR}{NS} \tag{4}$$

3.2. Overload rule

Overload rule is used to identify that whether the overload is caused when the task is under processing. For simplification, we apply 1 and 0 represent the overload happening and does not happening, respectively. The allocation to overload rule is based on the request. If the most submitted jobs are CPU intensive, we need both the CPU utilization rate and average load to judge the node is whether overloaded. If jobs are memory Intensive, we need take the utilization rate of memory into consideration. If the jobs have heavy IO activities, the utilization rate of Internet should be considered into the overload rule. This paper has considered both the CPU memory and the IO factors. All these observed value can be obtained by the running log of observed nodes.

3.3. The acquisition to weights based on the Logistic regression model

1) Logistic regression model

Logistic regression is probabilistic and nonlinear regression model. For n independent eigenvectors $x(x_1, x_2, ..., x_n)$, assuming the conditional probability $P(y = 1 \mid x)$ is the probability that event x happens according to the observed quantity. Thus the Logistic regression model can be represented by:

$$P(y = 1 \mid x) = \pi(x) = \frac{1}{1 + e^{-x}} \tag{5}$$

where $\frac{1}{1 + e^{-x}}$ is called as Logistic function. We set $x = w_0 + w_1 x_1 + w_2 x_2 + ... + w_n x_n$, w_0 denotes the intercept, and $w_1, w_2, ..., w_n$ mean slope. If we have the condition x, the probability that event y dose not happen is:

$$P(y = 0 \mid x) = 1 - P(y = 1 \mid x) = \frac{1}{1 + e^{x}} \tag{6}$$

Thus the ratio between the happening and un-happening is:

$$odds = \frac{P(y = 1 \mid x)}{P(y = 0 \mid x)} = e^{x} \tag{7}$$

This ratio is names as the Odds of Experiencing an Event. For convenience, we rewrite it as odds.

2) The acquisition to weights

For the task T (RW, TW and DL), the acquisition to weights w_0, w_1, w_2, w_3 refers to the learning of Logistic regression model [7]. Giving m observed samples from the running log, which respectively denote as $y_1, y_2, \ldots, y_m$, the probability that the overload happens is $p_i = P(y_i = 1 \mid x_i)$.Otherwise, the probability that the overload does not happen is $P(y_i = 0 \mid x_i) = 1 - p_i$. Therefore, the probability we get one observation is $P(y_i) = p_i^{y_i}(1 - p_i)^{1-y_i}$. As all the observed samples are independent each other, the simultaneous distribution is the product of all the marginal distribution. The likelihood function is:

$$L(w) = \prod_{i=1}^{m} [\pi(x_i)]^{y_i} [1 - \pi(x_i)]^{1-y_i} \tag{8}$$

Maximum likelihood estimation is aimed to obtain w_0, w_1, w_2, w_3, which can make the $L(w)$ get maximal value. We take the logarithm to $L(w)$:

$$\ln L(w) = \sum_{i=1}^{m} (x_i y_i - \ln(1 + e^{x_i})) \tag{9}$$

Vector $w = w_0, w_1, w_2, w_3$, which makes the $L(w)$ maximal, and $x_i = w_0 + w_1 x_{i1} + w_2 x_{i2} + w_3 x_{i3}$.

We Seek partial derivative for the likelihood function mentioned above:

$$\frac{\partial \ln L(w)}{\partial w_k} = \sum_{i=1}^{m} x_{ik}[y_i - \pi(x_i)] \tag{10}$$

We apply gradient rise to further handle Eq. (10), thus we have:

$$w_k = w_k + \alpha \frac{\partial \ln L(w)}{\partial w_k} = w_k + \alpha \sum_{i=1}^{m} x_{ik}[y_i - \pi(x_i)] \tag{11}$$

According to the functions above, we initialize the vector $w = w_0, w_1, w_2, w_3$ as random value and start to iterate until we get the given precision.

4. Experiment

In order to test the effectiveness of the proposed method, we construct the Hadoop cloud computing platform provided by the health information center in Nanjing. Specific configurations include 8 PC, 1 JobTracker and 7 TaskTracker. The CPU is Inter Core i3-4150 with 3.5GHZ, memory is DDR3-1600 with 4GB, and the hard disk is 500G with 7200 r/min. We adopt the Ubuntu 14.04.3 LTS as the operating system. The core is Linux 3.13.0, the java environment is jdk-6u24-linux-i586, and the Hadoop is 2.4.0. The Eclipse 3.4.1 is used as the development tool and the TP-Link interchanger with 16 mouth is applied as the network equipment.To observe the performance of different scheduling algorithms, we employ the representative WordCount as benchmark test program. The tested data is from RandomTextWrite in Hadoop. The Job configuration is listed in Table I. The job interval is set 15 sec. The block size is set as 32M.

Table I. The parameters of job

Job	Map number of tasks	The amount of data
Job1	4	128
Job2	8	256
Job3	16	512
Job4	32	1024
Job5	64	2048

1) The optimization experiment to data locality

We take the FIFO as benchmark scheduler and modify the proposed configuration optimization scheduling algorithm in the programmable interface of TaskScheduler. The obtained data from the WordCount application is visualized by OriginPro 8, which is showed in Fig. 1.

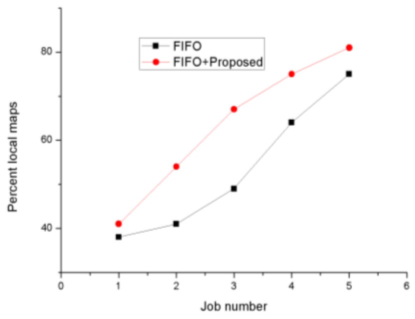

Fig. 1: The comparison of data locality.

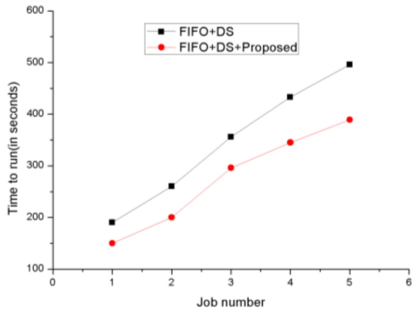

Fig. 2: The comparison of running time.

It can be seen from Fig. 1, when the amount of data is too large or too small, the earnings of FIFO is much weak. One reasons is that when data quantity is small, the effect of learning method to the scheduling impact is weak. The other reason is that when data quantity is large, the data locality of FIFO is already strong, thus the optimization is meaningless. When the data quantity is between 256M-1024M, that is the number of tasks is 8-32, the data locality is improved about 15% averagely, and the maximum is 18%.

This paper also take Delay Scheduling Algorithm as the optimized object for the experiment. As DS greatly increase the locality at the expense of the fair principle, our proposed method do not show the improved degree. However, for the situation strict with fair principle, the optimization of our proposed method is meaningful. As there are many factors when improving the MapReduce, we need to consider from different views in different situations.

2) The optimization experiment to running time

It can been from Fig. 2, as the overload of our method is equipped with learning ability, our method can keep adjusting when we optimize for DS. When the data quantity is increasing, the running time is relatively decreasing to DS. The average decreased time is about 70 sec, and the maximum value is about 100 sec in 2048M.

5. Conclusion

As existing scheduling algorithms are lack of concern to locality task, we introduce Logistic regression model to combine with Hadoop for the optimization of locality task. The experimental results demonstrate that the optimization algorithm based on machine learning does not improve the locality of map task, but can also obtain more efficient running time. The effectiveness of our paper is certainly meaningful to the large-scale data centers and cloud computing centers in the age of Internet plus. In future work, several other feature vectors which are influencing the priority will be considered and also should pay attention to some other performance such as load balancing and resource utilization.

6. Reference

[1] Fischer M J,Su Xueyuan,Yin Yitong. Assiging tasks for efficiency in Hadoop:extended abstract[C]//Proc of the 22nd ACM Symposium on Parallelism in Algorithms and Architectures,2010:30-39.

[2] Jin Jiahui,Luo Junzhou,Song Aibo,et al.BAR:An efficient data locality driven task scheduling algorithm for cloud computing[C]//Proc of 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. Washington DC:IEEE Computer Society, 2011:295-304.

[3] Zaharia M,Borthakur D,Sarma J S,et al.Delay scheduling:a simple technique for achieving locality and fairness in cluter scheduling[C]//Eurosys'10:Proceedings of the 5th European Conference on Computer System,2010:265-278.

[4] Zhang Xiaohong,Zhong Zhiyong,Feng Shengzhong, et al. Improving data locality of mapruduce by scheduling in homogeneous computing environment[C]// IEEE 9th International Symposium on Parallel and Distributed Processing with Applications(ISPA), 2011:120-126.

[5] Gu Yu, Zhou Liang, Ding Qiulin. Research of Three- Queue Scheduling Algorithms Based on Priority[J].Computer Science, 2011, 38(10a): 23-256.

[6] S.Suresh, N.P.Gopalan. An Optimal Task Selection Scheme for Hadoop Scheduling[C]//International Conference on Future Information Engineering, IERI Procedia 10, 2014: 70 – 75.

[7] Shi C. J., Zhang M. M. The analysis to logistic regression model. *Computer Aided Engineering*, 2005, 14(3): 74-78.