A Game Theory Based MapReduce Scheduling Algorithm

Ge Song[1], Lei Yu[2], Zide Meng[3], Xuelian Lin[4]

Abstract. A Hadoop MapReduce cluster is an environment where multi-users, multi-jobs and multi-tasks share the same physical resources. Because of the competitive relationship among the jobs, we need to select the most suitable job to be sent to the cluster. In this paper we consider this problem as a two-level scheduling problem based on a detailed cost model. Then we abstract these scheduling problems into two games. And we solve these games in using some methods of game theory to achieve the solution. Our strategy perfect fit the multi-level nature of the scheduling; and in using game theory, we can improve the utilization efficiency of each type of resources of the cluster, and we can also avoid the unnecessary transmission of data, finally find a balance between transmission and waiting.

Keywords: Multi-Level Scheduling, Task Assignment; Resource Utilization Efficiency; Bidding Model; The Hungarian Method; Game Theory; MapReduce; Hadoop

1 Introduction

MapReduce [1] is a programming model designed by Google for processing large scale

[1] Ge Song(✉)
Ecole Centrale de Pekin
Beijing University of Aeronautics and Astronautics, Beijing, China 100191
e-mail: sophiesongge@gmail.com
[2]Lei Yu
Research Institute of Beihang University in Shenzhen
Ecole Centrale de Pekin
Beijing University of Aeronautics and Astronautics, Beijing, China 100191
e-mail: yulei@buaa.edu.cn
[3]Zide Meng, [4]Xuelian Lin
School of Computer Science and Engineering
Beijing University of Aeronautics and Astronautics, Beijing, China 100191
e-mail: [3]mengzide@act.buaa.edu.c, [4] linxl@act.buaa.edu.cn

data sets parallel and distributed. It provides a simple and powerful way to let the programs run in a distributed environment automatically. Apache Hadoop [3] is an open source implementation of MapReduce. It consists of a distributed file system named HDFS (Hadoop Distributed File System), and a MapReduce programming framework. Due to the simplicity of the programming model and its elastic scalability and fine-grained run-time fault tolerance [2], Hadoop is popular among the commercial enterprises, the financial institutions, the scientific laboratories and the government organizations.

The Hadoop clusters take a master-slave design pattern. The Masters are NameNode (for HDFS) and JobTracker (for MapReduce). NameNode is in charge of maintaining the namespace of HDFS, managing the mapping relationship of filename and blocks, interacting information with DataNode, and so on. JobTracker is in charge of scheduling the tasks to the nodes, and monitoring them to see whether it is finished or not. The Slaves are DataNodes (for HDFS) and TaskTrackers (for MapReduce). DataNodes are the nodes which will store the data; and TaskTrackers are the nodes in which the data will run. Generally, a Hadoop job will be scheduled according to 3 steps: user level, job level and task level. And usually, in a Hadoop cluster 3 types of resources will be consumed, they are: CPU, Disk and Network.

This paper analyses the advantages and the shortcomings of the existed Hadoop scheduling strategies, and designs a new scheduling algorithm. This algorithm firstly meets the multi-level nature of the Hadoop scheduling environment, to minimize the average waiting time per-user. Then it finds a balance point between “transfer data” and “wait” when running map tasks, to optimize the global cost of all the map tasks.

2 Related Work

2.1 The Analysis for the Existing Hadoop Scheduling Algorithms

In order to providing convenience to the users, the scheduler is designed as a pluggable model, so that users can design their own scheduler under their needs. And at the same time, Hadoop framework also integrates 3 schedulers to be invoked by users.

Firstly, the scheduler by default uses the FIFO (First In First Out) strategy. In this strategy, all the jobs will be submitted to a single job queue, and then JobTracker will select the first job sent to the queue to be executed in accordance with their priority. Capacity Scheduler [4] and Fair Scheduler [5] are also integrated in a Hadoop framework. Both of those support multi-queue scheduling policies. Capacity Scheduler

wants to limit the resources given to the jobs submitted by the same user, in order to prevent the resources being exclusive. Fair Scheduler's purpose is to make sure that the resources will be allocated fairly to each job.

At the same time, there also appear many schedulers proposed by the users aiming at different application scenarios. The most common schedulers are the following three types:

1. The first type leads by [6] and [12] who want to ensure the data locality. Data locality means placing tasks on the nodes that contain their input data. [6] comes to a conclusion that there is a conflict between fairness and data locality. In order to solve this problem, [6] gives an algorithm called Delay Scheduling. Its mean idea is: instead of launching a job according to fairness whose tasks cannot be run locally, it chooses to let it wait for a small amount of time, and let other jobs launch their tasks locally. But this paper did not give a balance between "waiting" and "transmission" from the point of view of the global cluster.

[7] also aims to avoid unnecessary data transmission. Its method is: to assign tasks to a node, local map tasks are always preferred over non-local map tasks, no matter which job the task belongs to. This method can guarantee the Data Locality in a certain extent, but it greatly impairs the levels of Hadoop scheduling.

2. The second type is deadline scheduling as [8]. Its purpose is to allocate the appropriate amount of resources to the job so that it meets the required deadline. Usually, this kind of scheduling policy always applies to a cost model to estimate the complete time of the task. But they tend to use a simple model which cannot dynamically estimate the complete time of each task. That strongly affects the efficiency of the scheduling algorithm.

3. The third type is called performance-driven scheduling. [9] wants to dynamically predict the performance of the tasks, and uses the prediction to adjust the allocation of resources for the jobs. This can help to allocate as needed and avoid wasting resources. But, this kind of scheduling relies on an important part ---- the cost model. However the cost model they used cannot exactly predict the performance of the tasks which reduces a lot of validity of the scheduling strategy.

2.2 Cost Model

The cost models are usually used in task assignment and resource allocation. There are 3 common kinds of cost models. The first one doesn't consider the steps within a task, they simply think that the complete time of a local task is 1, and that of a remote task is 3. This kind of cost model is usually used in approximation algorithms, such as [13]. The second one is called a "non-accurate" model. This kind of model usually assumes

that the cluster is isomorphic, and the running times of the tasks are equal. They use the performance of an already run task or the history trace to predict the execution time of a new task. Deadline scheduling often uses this type of cost model.

The third kind of model divides the cost as Disk IO, CPU and network transfer cost and so on. [10] gives a method to quantitatively describe each kind of cost of a task. We can use [10] to effectively and accurately predict the performance of a task. The scheduling algorithm in our paper is based on a prediction of the costs of the tasks according to [10].

2.3 Game Theory Used for Scheduling in Cloud

Game Theory studies the balance when the decision-making bodies interact among each other under a related constraint. It solves an optimization problem of vectors which contains the target vector and the strategy vector. It can be divided into cooperative game and non-cooperative game based on the behavior of the decision makers. It is originally used in economics, but recently its approaches are always successfully used in resource allocations in cloud environment [14].

The scheduling problems of a distributed environment involve complicated optimization requirements. In solving this kind of problems, traditional methods usually combine the individual optimizations as a solution. And it only cares about the individual rather than the entirety.

For a scheduling system under a Hadoop environment, the multi-level nature as well as the diversity of the optimization objectives makes a game theory method obviously better to solve this problem.

3 Problem Statement

After researching, we found the Hadoop scheduling environment has two natures below:

1. Multi-Level: We can schedule the Hadoop jobs by 3 levels: user level, job level and task level.

2. Consuming of multi-resource: Running a Hadoop job will consume 3 kinds of resources: CPU, Disk IO and Network IO. Other scheduling strategies only consider the allocation of CPU resources. But through some benchmarks we found that sometimes the random Disk IO will give a greater impact for the completion time of a job.

So we hope to design a scheduling algorithm, which can meet the following characteristics:

1. It can meet the needs of multi-level: a) take care of the fairness among each user. b) let the "short jobs" run earlier. c) avoid the jobs from waiting too long. b) and c) want to reduce the average waiting time of each jobs, especially the average waiting time per-user.

2. It should improve the utilization of the multi type of resources.

3. It has to find a balance point between "transfer" and "wait".

To fit the purpose above, we design a 2 level scheduling algorithm: Level 1 Jobs Scheduling (with the information of user) and Level 2 Tasks Scheduling. The scheduling schema is shown as Fig. 1. Then we will present these 2 level scheduling algorithms respectively.

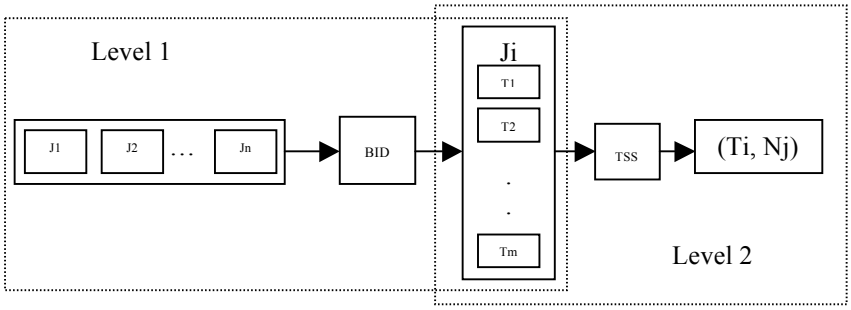

Fig. 1 Scheduling Schema

3.1 Level 1: Jobs Scheduling (with the Information of User)

To simplify the problem we choose to add the user information into the definition of a job, so that we do not have to consider about the user level scheduling separately.

Definition 1: A MapReduce Job Queue (Q):

A $Q(j_1, j_2, \dots, j_q)$ is a vector which represents the set of submitted jobs. j_i means each job in Q.

Let q=|Q|, which means the number of jobs in the current Q.

Let j_i_user be the user who submits j_i.

Q dynamically changes when a job comes to Q or when a job leaves Q.

Definition 2:

Total User Time: the sum of the time of all the jobs submitted by this user.

Wait Time: the time that the job has waited after it was submitted

Estimate Exec Time: the execution time of the job estimated by the cost model

Total User Time can represent the occupancy time of the cluster by a given user. Wait Time is used to reduce the waiting time of the jobs. Estimate Exec Time is

predicted by a cost model presented in [10].

According to the analysis above, we can give the definition of the "bid" given by a job to represent its priority of being executed.

Definition 3: Bid

$$\text{bid} = \frac{1}{\text{TotalUserTime}} \times \frac{\text{WaitTime}}{\text{EstimateExecTime}}$$

$\frac{1}{\text{TotalUserTime}}$ is designed for the fairness. And $\frac{\text{WaitTime}}{\text{EstimateExecTime}}$ is used to reduce the average waiting time of the jobs.

Game 1: Hadoop Job Scheduling Game

The target of this game is to choose a suitable job to be executed next.

To achieve this target we choose the bidding model, which is a dynamic non-cooperative game. There is a competitive relationship among the jobs. They compete for resources. Every job wants to be executed as soon as possible. The bid defined in definition 3 can reflect the trend of our scheduling purpose. So we let each job in Q submit a bid to JobTracker according to definition 3, then the job with the highest bid will be sent to the cluster and run.

3.2 Level 2: Tasks Scheduling

Tasks scheduling is a cooperative game because we want to make the execution cost of the job minimum which means we should take the tasks as an entirety. This doesn't mean to make each task run as soon as they can. Our goal is to reduce the global complete cost of all the map tasks. This cost means the total cost this set of map tasks occupy the cluster. So to reduce this cost means to reduce the depletion of the resources of the cluster.

To achieve this goal, we give some definitions below:

Definition 4: A MapReduce Job (MR-Job)

A MR-Job is a pair (T, N) where T is a set of tasks, and N is a set of nodes.

Let m=|T|, and n=|N|, which means, m is the number of tasks in T, and n is the number of nodes in N.

Definition 5: A tasks scheduling strategy (TSS)

A TSS is a function X: T→N that assigns each task to a node n.

We define also a x_{ij}, if X(j)=i, then x_{ij}=1, if not, x_{ij}=0. Which means, if assign task j to node i, then x_{ij}=1, if not, x_{ij}=0.

Definition 6: A Cost Function C:

Let C(i, j)=(C, D, N) be the cost vector of running task j on node i, with C, D, N represent the cost of CPU, Disk I/O, and Network I/O separately.

Let c_{ij} = c×C + d×D + n×N, where c, d, n are the weights of C, D, N. These

weights will vary because of the type of the jobs. If the job is compute-intensive, then c will be bigger. If it is a data parsing job, then d and n will be bigger.

We define c_{ij} the cost of running task j on node i, we call the matrix c_{ij} a cost matrix. And the cost matrix c_{ij} is called the parameter matrix for Hungarian Method.

Definition 7: Total cost under a TSS:

$$Z = \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} \times x_{ij}$$

Game 2: Hadoop Tasks Scheduling Game

In this game we have to find an assignment for map tasks to minimize the execution cost of the entire map tasks. This is an assignment problem, which is a static cooperative game. In an assignment problem, we have m tasks, and they will be completed by n nodes. And we have to find a task scheduling scheme that let the total cost be the least. If we transfer this target into mathematics, it means to find a X (t), that minimize Z, where X(t) can represent an assignment.

We choose Hungarian Method [11] to solve this problem. And we translate this problem into mathematics:

$$\min z = \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij} \quad (1)$$

$$s.t = \sum_{i=1}^{n} x_{ij} = 1 \quad i = 1,2,\dots,n \quad (2)$$

$$\sum_{j=1}^{m} x_{ij} = 1 \quad j = 1,2,\dots,n \quad (3)$$

$$x_{ji} = 0 \text{ or } 1 \quad (4)$$

(2) means each task can only be executed by one node. And (3) means each node can only execute one task at one time. These 2 equations are the constraints for this assignment problem.

Theorem 1:

According to the nature of the matrix multiplication, z won't change if we plus or minus a constant to all the elements in a row or in a column of c_{ij}. Which means that there won't be any differences to the assignment if we plus or minus a same constant to a row or a column in c_{ij}.

PROOF.

Suppose $c'_{ij} = c_{ij} \pm (u_i + v_j)$, then,

$$Z' = \sum_{i=1}^{n} \sum_{j=1}^{m} c'_{ij} x_{ij} = \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij} \pm \left(\sum_{i=1}^{n} \sum_{j=1}^{m} u_i x_{ij} + \sum_{i=1}^{n} \sum_{j=1}^{m} v_j x_{ij} \right)$$

x_{ij} is a vector with the value of 0 or 1, so the above equation is equal to:

$$\sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij} \pm \left(\sum_{i=1}^{n} u_i + \sum_{j=1}^{m} v_j \right) = \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij} + K = Z + K$$

K is a constant. So the x_{ij} which makes Z reach the minimum will also makes Z' minimum.■

The Hungarian Method depends on this theorem. So we can solve the problem as follows:

Step 1: Input a cost matrix c_{ij}

Step 2: Find the smallest values in each column, for each element in this column,

minus this smallest value, repeat for each row.

Step 3: Check in each column, mark the first 0 in this column, the other 0 will be deleted

Step 4: Check whether the number of 0 is equal to that of tasks. If equal, break. If not, use the least lines to cover all the 0, then delete the elements being covered.

Step 5: Find the minimum value in the elements non-deleted, and for all the rows which contains the elements non-deleted, minus this minimum. If there comes a negative element, plus this element with this minimum. A new matrix will be formed after doing this. Come to step 3 for this new matrix.

4 Experiment Stuying

We intend to use simulations to verify the effectiveness of our algorithm.

Average Waiting Time per User:

We generate a set of jobs, with the number from 30 to 200, and schedule them with both FIFO policy and BID policy proposed in our paper. Then we record the average waiting time per job as Fig. 2:

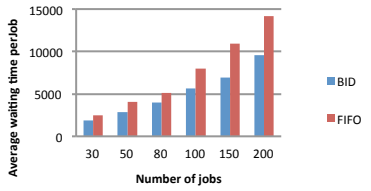

Fig. 2 compare the bid method with FIFO for the average waiting time per job

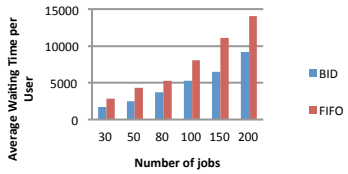

Fig. 3 compare the bid method with FIFO for the average waiting time per user

And the average waiting time per user is shown as Fig. 3:

From these 2 figures we can see that both the average time per job and per use of Bid method given in this paper are shorter than those of FIFO. And with the increase in the numbers of jobs, this gap becomes more and more obvious.

Data Locality Rate:

We suppose that the number of tasks run locally is l, and the number of tasks run remotely is r. Then we can define a data locality rate as: $DLR = \frac{l}{r}$. This rate describes the degree of tasks' localization. In this simulation we suppose there are 3 replicas for each input split, stored in 3 nodes separately. We execute 5 tasks and 10 tasks separately in the cluster with the number of nodes varies from 10 to 60. Compared with

the scheduling strategy in Hadoop by default, we can see the data locality rate as Fig. 4:

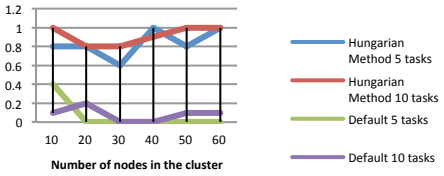

Fig. 4 Data Locality Rate

We can see the method we proposed is much better than that by default in Hadoop, especially when the number of nodes is much bigger than that of tasks.

Global Tasks Complete Time:

In this part, we want to measure the global complete time of all the tasks for a job which contains 5 map tasks (Fig. 5), and another job which contains 10 map tasks (Fig. 6).

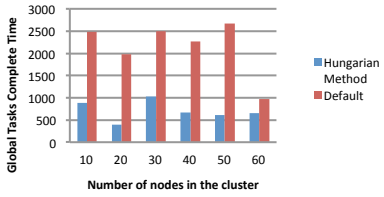

Fig. 5 Global complete time for 5 tasks

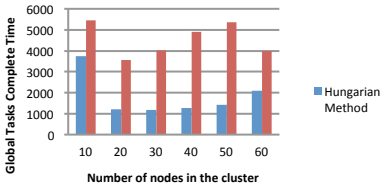

Fig. 6 Global complete time for 10 tasks

From these 2 pictures we can tell that our method will take much less time for complete all the tasks than the scheduling strategy by default.

5 Conclution

Through analyzing the performance of the scheduler model of Hadoop, we find some problems and give a game theory based method to solve these problems. We divide a Hadoop scheduling problem into 2 steps----job level and task level. For the job level scheduling we choose to use a bid model, and we define this bid in order to guarantee the fairness and reduce the average waiting time. Then for tasks level, we change this problem into an assignment problem and use the Hungarian Method to optimize this

problem. At last we do some simulation experiences to prove the efficiency of our algorithm.

For the further research, we want to improve the performance of our scheduling strategy in the following aspects: 1. Prove this algorithm in mathematics in using the methods of game theory. 2. Do experiences in a real cluster. 3. Automatically give the weights of the costs. Etc.

Acknowledgement. This work was supported by the National High Technology Research and Development Program of China (No.2011AA010502) and the National Science & Technology Pillar Program (2012BAH07B01)

References

1. Jeffrey Dean, Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. Proceedings of the Sixth Symposium on Operating System Design and Implementation (San Francisco, CA, Dec. 6-8). Usenix Association(2004)
2. Dawei Jiang, Beng Chin Ooi et. Al. The Performance of MapReduce: An In-depth Study. Proceedings of the VLDB Endowment 2010, 3(1)(2010)
3. Apache Hadoop. http://hadoop.apache.org
4. Capacity Scheduler: http://hadoop.apache.org/common/docs/r0.20.2/capacity_scheduler.html
5. Fair Scheduler: http://hadoop.apache.org/mapreduce/docs/r0.21.0/fair_scheduler.html
6. Matei Zaharia, Dhruba Borthakur et. al. Delay Scheduling: A Simple Technique for Achieving Locality and Fairness in Cluster Scheduling, EuroSys' 10
7. Chen He, Ying Lu et. al. Matchmaking: A New MapReduce Scheduling Technique, CloudCom '11
8. Abhishek Verma, Ludmila Cherkasova et. al. ARIA: Automatic Resource Inference and Allocation for MapReduce Environments, ICAC' 11
9. Jorda Polo, David Carera et. al. Performance-Driven Task Co-Scheduling for MapReduce Environments, NOMS' 2010
10. Xuelian Lin, Zide Meng et. al. A Practical Performance Model for Hadoop MapReduce, Cluster (2012)
11. H. W. Kuhn. The Hungarian Method for the Assignment Problem, Bryn Mawr College
12. Matei Zaharia, Dhruba Borthakur et. al. Job Scheduling for Multi-user MapReduce Clusters, EECS Department, University of California, Berkeley, Tech. Rep.(2009)
13. Michael J.Fischer, Xueyuan Su et. al. Assigning Tasks for Efficiency in Hadoop, SPAA '10
14. Samee Ullah Khan, Ishfaq Ahmad. Non-cooperative, semi-cooperative, and cooperative games based grid resource allocation, Parallel and Distributed Processing Symposium, vol. 0, pp.101(2006)