Two-Stage Job Scheduling Model Based on Revenues and Resources

Yuliang Shi[1(✉)], Dong Liu[2], Jing Hu[1], and Jianlin Zhang[1]

[1] School of Computer Science and Technology, Shandong University, Shunhua Road 1500, High-tech Development Zone, Jinan, Shandong, China
shiyuliang@sdu.edu.cn, hujing_ll25@163.com, dareway_zjl@126.com

[2] Marketing Department, State Grid Chongqing Electric Power Company Marketing Department, Zhongshan Three Road 21, Yuzhong District, Chongqing, China
LD8803@163.com

Abstract. In the big data platform, multiple users share the resources of the platform. For platform providers, it is a problem to be solved urgently that how to multi-user jobs are scheduled efficiently to take full advantage of the resources of the platform, get the maximum revenue and meet the SLA requirements of the users. We research the project of job scheduling for MapReduce framework further. The paper proposes a two-stage job scheduling model based on revenues and resources. In the model, we design a scheduling algorithm of the maximum revenue (SMR) based on the latest start time of the jobs. The SMR algorithm ensures that the jobs which have larger revenues can be completed before the deadlines of the jobs, and then providers can gain the largest total revenue. Under the premise of ensuring the maximum revenue, a sequence adjustment scheduling algorithm based on the maximum resource utilization of the platform (SAS) is developed to improve the resource utilization of the platform. Experimental results show that the two-stage job scheduling model proposed in this paper not only realizes the maximum revenue of the provider, but also improves the resource utilization of the platform and the comprehensive performance of the platform. What is more, the model has great practicability and reliability.

Keywords: Big data · Job scheduling · Revenue · Resource utilization

1 Introduction

In recent years, with the vigorous development of cloud computing and Internet technology, the data show an explosive growth mode to make big data quietly come. Traditional data processing technology and tools are unable to meet the requirements of data processing in the new era, so the big data platform emerges at a historic moment. The big data platform supports a variety of computing frameworks that can serve multiple users simultaneously. However, it is an urgent problem for platform service providers that the mixed jobs of multiple users are reasonably scheduled so as to meet the requirements of resources utilization, the SLA of users and the maximum revenue.

X. Shi et al. (Eds.): NPC 2017, LNCS 10578, pp. 37–48, 2017.
DOI: 10.1007/978-3-319-68210-5_4

At present, many researchers have studied the problem of job scheduling in big data platform and put forward a lot of solutions. A resource configuration optimization model based on deadline estimation for the Pig job is proposed [1]. The model eliminates the non-deterministic problem when Pig program executes jobs concurrently. However, the model does not consider the revenue of the platform. In [2, 3], researchers study the job scheduling project on basis of MapReduce framework. While taking into account deadline constraints and resource allocation, they do not consider the revenues of the platform and resource utilization. The proposed scheduling algorithms [4, 5] focus on the job deadlines. Although these algorithms are not suitable for our study, they provide a great guide. Liu et al. [6] propose a priority scheduling algorithm to divide the computing capacity of each node into a front-end virtual machine layer and a background virtual machine layer. The algorithm balances the workload of the platform, makes full use of the CPU resources, improves the execution efficiency of the job and shortens the response time of the job. However, it still does not consider the revenue of the platform. Koutsandria et al. [7] first investigate the problem of efficient resource allocation strategies for time-varying traffic, and propose a new algorithm, MinDelay, which aims at achieving the minimum service delay while taking into account the revenue of the providers.

To sum up, the above researches have been studied deeply in different constraints and different backgrounds for the job scheduling project and a series of achievements have been made. However, the methods in these achievements do not solve the problems of our study. We propose a two-stage job scheduling model based on revenues and resources in MapReduce framework and the main contributions of this paper are summarized:

- We design a scheduling algorithm of the maximum revenue (SMR) based on the latest start time of the jobs. According to the deadlines of the jobs and the revenue rate, SMR pre-allocates the resources of the platform to the jobs and adjusts the allocation result to make the provider gain the maximum revenue.
- According to SMR, we propose a sequence adjustment scheduling algorithm based on the maximum resource utilization of the platform (SAS). The job sequence on basis of the maximum revenue is adjusted to realize the maximum resource utilization under the premise of the maximum revenue.

The rest of the paper is organized as follows. In Sect. 2, we discuss the related work of job scheduling. Section 3 presents relevant definitions and the design of two-stage job scheduling model. The model is described detailedly in Sect. 4. In Sect. 5, the experimental results of the scheduling model are given and analyzed. Section 6 concludes the paper.

2 Related Work

Job scheduling in the big data platform is crucial to the optimization of the platform performance. In order to improve the efficiency of the job execution and optimize the performance of the platform, the researches [8–10] propose data placement strategy and job scheduling algorithm based on the minimum data transmission time to reduce the

data transmission time and improve the efficiency of the job execution. However, the revenue is not considered in the algorithms. The key factors affecting the availability requirement of the parallel task and the guarantee of the resource availability are analyzed, and a parallel task scheduling algorithm based on usability perception is proposed [11]. Although the methods [12–14] optimize the performance of the platform, they do not take into account the revenues of the platform. These researches [15–17] optimize the performance of the platform by reasonably scheduling jobs, but the revenue and the resource utilization are not considered. In order to improve the utilization of the network and reduce the completion time of the job, a job-aware priority scheduling algorithm is proposed by monitoring the application layer [18]. The algorithm not only achieves network load balancing, but also improves the execution efficiency of the job. Kumar et al. [19] propose a scheduling algorithm based on perceptual and heterogeneous cluster to improve the resource utilization, whereas they do not consider the revenues.

In [20], job scheduling is studied from the perspective of resource allocation, and a flexible resource allocation algorithm is proposed to enable the job to be completed and consume the minimum computing resources before the deadline. The algorithm provides a fine guide for the resource scheduling. An intelligent job scheduling and workload balancing mechanism is proposed to realize the application performance with the least resources [21]. The framework does not guarantee the maximum resource utilization. The research [22] solves the issue of the maximum revenue, but there are some limitations. The method does not take into account the situation of parallel execution for multiple jobs. However, these methods do not take into account the maximum revenue and the maximum resource utilization at the same time. Therefore, this paper considers the constraints of the deadline, the maximum revenue and the maximum resource utilization and proposes a two-stage job scheduling model. The model not only meets the requirements of the deadlines, but guarantees to maximize the revenue and the resource utilization.

3 Two-Stage Job Scheduling Model

3.1 Basic Definitions

In order to facilitate the description of the job scheduling model, we give some basic definitions and formulas in this section.

Definition 1. *Total Number of Computing Resources (TR): the number of all Containers in the platform.*

The big data platform that one master node and N_{dn} workers. Each node is configured with c cores and m GB RAM. Each Container is configured with c_c cores and m_c GB RAM. The total number of resources of the platform:

$$TR = min(c/c_c, m/m_c) \times N_{dn}. \tag{1}$$

Where $min(c/c_1, m/m_1)$ is the maximum number of Containers for each node.

Definition 2. *A Set of Submitted Jobs: when the jobs are submitted by multiple users signing the SLAs with the provider to the big data platform, a set of submitted jobs is generated. It is expressed as* $J = \{j_1, j_2, \ldots, j_n\}$*, where n is the number of the submitted jobs.*

We focus on the job scheduling in the isomorphic cluster, so we set identical nodes that have identical hardware configuration and performance. What is more, in this paper, we do not consider data skew so that we think the running time of every map task or reduce task is same by default. We pay attention to the execution time, required resources, the deadline and the revenue for each job. Any job *j* is expressed as *j = (ms, rs, mt, rt, dl, rf(t))*. Where *ms* is the number of map tasks, *rs* is the number of reduce tasks, *mt* is the average execution time of map tasks, *rt* is the average execution time of reduce tasks, *dl* is the deadline of the job and *rf(t)* is the revenue function of the job. The revenue function is following:

$$rf(t) = \begin{cases} a, & t \leq j.dl \\ b, & t > j.dl \end{cases}. \tag{2}$$

Where *a* is the revenue gained for the provider when the job is completed before the deadline and when the job is not completed on time, *b* is positive value as the revenue that is less than *a*. If *b* is negative value, the provider compensates the user for the loss.

Definition 3. *The Total Revenue Function: the job* j_i *has* $j_i.dl$ *and* $j_i.rf(t)$*. The actual completion time of* j_i*,* $j_i.end$*, may be more than* $j_i.dl$ *or less than* $j_i.dl$*. For all jobs, the total revenue function is following:*

$$R = \sum_{i=1}^{n} j_i.rf(j_i.end). \tag{3}$$

3.2 Scheduling Model Design

In this section, we design the architecture of job scheduling model. In the first part of Fig. 1, the processing of job scheduling is divided into two stages. The first stage is a scheduling algorithm of pre-allocation resources based on the latest start time and maximum revenue. The second stage is a sequence adjustment scheduling algorithm based on the maximum resource utilization of the platform. In the first stage, the submitted jobs generate a set of jobs awaiting to be scheduled. Then, the jobs are scheduled firstly by pre-allocating the resources of the platform based on the latest start time and maximum revenue. The result of the first stage is that a primary sequence of scheduled jobs is got by SMR. In the second stage, the primary sequence is adjusted by SAS according to the remaining resources. The final result of the model is that generating a job execution sequence to make the provider gains the maximum revenue and improve the resource utilization. In the second part, the job scheduler launches scheduled jobs based on the optimal start time of the jobs. The resource scheduler allocates resources to launched jobs and the jobs gain resources from different nodes.

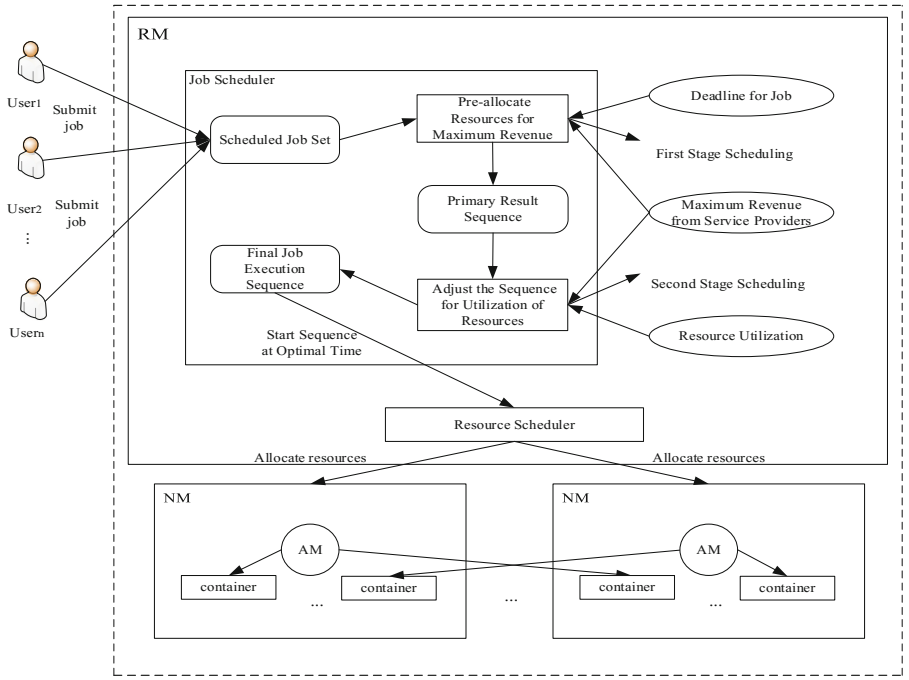

Fig. 1. The architecture of two-stage job scheduling model

4 Model Implementation

4.1 SMR Algorithm

In the section, it is showed that the implementation of SMR algorithm and the definition of equations needed in the SMR algorithm.

Definition 4. *Initial Latest Start Time $j_i.T_{ols}$: the job is just completed at the deadline when the resources are not competed by other jobs. If the deadline of j_i is $j_i.dl$, the initial latest start time is following:*

$$j_i.T_{ols} = j_i.dl - \left(\left\lceil \frac{j_i.ms}{M} \right\rceil \times J_i.mt + \left\lceil \frac{j_i.rs}{M} \right\rceil \times j_i.rt \right). \tag{4}$$

Where M is the available resources of the platform. $J_i.ms/M$ is the execution rounds of map tasks and $J_i.rs/M$ is the execution rounds of reduce tasks.

The job j_i is launched before $J_i.T_{ols}$ to ensure the job completed before the deadline of the job when there is no resource contention. The job j_i is not completed before $j_i.dl$ when j_i is launched at $J_i.T_{ols}$. Therefore, we should adjust the latest start time for jobs competing resources. To determine the time period for resource contention, we should pre-allocate the resources to all jobs based on the initial latest start time of the jobs. The total computing resources are counted at statistics time period. At different time period,

different states may arise. There are two states, normal load state and overload state. The time period in the overload state is called the overload time period. In the overload time period, the resources of the platform could not meet the resource requirements of the jobs so as to delay the jobs eventually.

For the overload time period, we should design an optimal adjustment strategy to guarantee the maximum revenue. A feasible adjustment strategy set is defined as follows:

Definition 5. *A Set of Feasible Adjustment Strategy: there is a set of pre-scheduled jobs during the overload time period, called as Ju. A minimum proper subset is selected from Ju, Js ⊆ Ju. The initial latest start time is advanced for all jobs of Js so that the overload state turns into the normal state. Js is called as a feasible adjustment strategy. A set of feasible adjustment strategy during a certain period of time is* $CL = \{Js_1, Js_2, \ldots, Js_m\}$.

The latest start time of all jobs, in Js_i , is advanced to the overload state changing to the normal state. The latest start time for all jobs is changed as follows:

$$j_i.T_{ls} = T_{cs} - \left(\left\lceil \frac{j_i.ms}{M} \right\rceil \times J_i.mt + \left\lceil \frac{j_i.rs}{M} \right\rceil \times j_i.rt \right). \tag{5}$$

Where $j_i \in Js_i$, and T_{cs} is the start time of the overload time period.

During the overload time period, we should select a feasible adjustment strategy from *CL*. To guarantee the maximum revenue, the feasible adjustment strategies are evaluated for the revenues. Main factors of the evaluation are following two aspects, the Evaluation of the revenue (*Sp*) and adjustment cost.

$$Sp = |a - b|. \tag{6}$$

Taking into account the two factors, the paper presents an evaluation function of the adjustment strategy based on the goal of the maximum revenue. The adjustment strategy of the minimum score is the optimal adjustment strategy. The evaluation function is following:

$$Js_i.pf = \frac{\sum\limits_{m \in Js_t} j_m.Sp}{\sum\limits_{m \in Ju} j_m.Sp} \times lastsize. \tag{7}$$

Where $\sum_{m \in Js_t} j_m.Sp$ is the sum of the revenue valuation for all jobs in the adjustment strategy $\sum_{m \in J_u} j_m.Sp$ is the sum of the revenue valuation for pre-scheduled jobs during the overload time period and *latesize* is the ratio of the remaining resources of the current time period and the total resources of the platform.

The SMR algorithm is outlined in Algorithm 1. The initial latest start time is calculated for each job and the resources are pre-allocated to submitted jobs according the initial latest start time (lines 1–3). The resources are counted for every time period based on the result of pre-allocation, which is called as *P_R* (line 4). If the overload time period is existing, the initial latest start time are adjusted for all jobs during the

overload time period (lines 5–21). The last overload time period is selected (line 6) and the set of pre-scheduled jobs is got during the overload time period (line 7). The adjustment strategies are evaluated by the evaluation function and the optimal adjustment strategy is selected (lines 8–13). The latest start time of the jobs in the adjustment strategy are adjusted and *P_R* is updated (lines 14–20). Looping through lines 6-20 until there is no overload time period in *P_R*.

Algorithm 1. SMR algorithm

Input: $J=\{j_1, j_2, \ldots, j_n\}$
Output: P_R //Resource preemption results
1. **for** each job $j_i \in J$ **do**
2. $j_i.T_{ols} = j_i.dl-(\lceil(j_i.ms)/M\rceil \times j_i.mt+\lceil(j_i.rs)/M\rceil \times j_i.rt)$;
3. Preempted resource for j_i at $j_i.T_{ols}$;
4. $P\_R=\{(P_1, R_1), (P_2, R_2), \ldots, (P_t, R_t)\}$; //Calculate resources required at each period
5. **while** $\exists(P_i, R_i) \in P\_R, R_i > TR$ **do**
6. Select the last $(P_n, R_n) \in P\_R, R_n > TR$;
7. *Ju* is equal to the jobs executed at P_n;
8. Get $CL= \{Js_1, Js_2, \cdots, Js_m\}$; $ZYCL= Js_1$;
9. $Js_1.pf=(\sum_{m \in Js1} j_m.Sp)/(\sum_{m \in Ju} j_m.Sp) \times lastsize$; $maxPf=Js_1.pf$;
10. **for** each $Js_n \in CL$ **do**
11. $Js_n.pf=(\sum_{m \in Jsn} j_m.Sp)/(\sum_{m \in Ju} j_m.Sp) \times lastsize$;
12. **if** $Js_n.pf > maxPf$ **then**
13. $maxPf=Js_n.pf$; $ZYCL= Js_n$;
14. **for** each $j_i \in ZXCL$ **do**
15. $j_i.T_{ls} = T_{cs}-(\lceil(j_i.ms)/M\rceil \times j_i.mt+\lceil(j_i.rs)/M\rceil \times j_i.rt)$;
16. **if** $j_i.T_{ls} > 0$ **then**
17. Preempted resource for j_i at $j_i.T_{ls}$;
18. **else**
19. Abandon j_i;
20. Update *P_R*;
21. return *P_R*;

4.2 SAS Algorithm

In the section, we should consider the pre-allocation of resources based on the SMR algorithm to ensure the jobs completed on time. To maximize the resource utilization of the platform, we evaluate the utilization of the computing resources using the rate of waste resources (W_{rr}). W_{rr} is the ratio between the non-reusable resources W_r after scheduling jobs and the sum of the currently used computing resources A_r. The smaller the waste resource rate, the larger the resource utilization of the platform currently. W_{rr} is following:

$$W_{rr} = \frac{W_r}{A_r} \tag{8}$$

The SAS algorithm is presented in Algorithm 2. A set of jobs E_j is found to be executed at T time and the required resources of E_j do not conflict with pre-allocated resources (lines 2–8). If E_j is not null, the job is selected that makes the waste resource rate minimize and the optimal start time of the job is T (lines 10–16). If E_j is null, T is set to the start time of the next time period in P_R (line 18). Looping through lines 3–20 until every job has an optimal start time.

Algorithm 2. SAS algorithm

Input: $J = \{j_1, j_2, \cdots, j_n\}$, P_R
Output: *ResultSet* //An execution sequence of jobs contains start time of each job

```
1.  while J=null do
2.    E_j=null;
3.    for each j_i∈J do
4.      N_P_R=P_R.remove(j_i);
5.      N_P_R=N_P_R.add(j_i, T);
6.      if ∀(P_i, R_i) ∈ N_P_R, R_i≤TS then
7.        Compute j_i.W_rr;
8.        E_j.add(j_i);
9.    if E_j=null then
10.     Sort jobs in E_j;
11.     Get the job j_l with the least W_rr;
12.     if j_l has advanced resources then
13.       P_R=P_R.remove(j_l);
14.     ResultSet.add(j_l, T);
15.     J.remove(j_l);
16.     P_R=P_R.add(j_l, T);
17.   else
18.     T is equal to the start time of next period in P_R;
19. return ResultSet;
```

After the second adjustment scheduling, we adjust the initial scheduling sequence to reset the start time for each job. As the result of the second adjustment and scheduling, the resource utilization of the platform becomes larger and the actual start time and the completion time of the job are advanced. Many of the jobs discarded due to insufficient resources in the SMR algorithm have opportunity to be executed newly. Therefore, the revenue for the provider may get greater.

5 Performance Evaluation

5.1 Experiment Setup

Platform Configuration. We experiment with the proposed algorithm in a big data platform based on MapReduce computing framework. The platform contains one master node and 20 workers that have identical configuration. The configuration information of the node is CPU 8 cores, 8 GB RAM, 1 TB hard disk, Red Hat

Enterprise Linux 6.5, and Hadoop 2.7.1. Each Container is configured with 1 core and 2 GB RAM so that each node has 4 Containers and the platform has 80 Containers.

Performance Indicators. In order to verify the effectiveness of the scheduling algorithm proposed in this paper, two - phase scheduling algorithm (TPS) is compared with FIFO and EDF Scheduler in the effect of different performances. We evaluate the algorithm using three indicators, the platform resource utilization (*PRU*), the job completion rate (*JCR*) and the total revenue (*PR*). The three indicators are as follows:

$$PRU = \frac{\sum_{j=1}^{k} RE\_j_i}{TR}. \tag{9}$$

Where RE_j_i is the occupied resources of the job i executed in the platform.

$$JCR = \frac{n}{N}. \tag{10}$$

Where n is the number of the jobs completed before the deadlines and N is the number of all jobs submitted by users.

$$PR = \sum_{j=i}^{n} a_j - \sum_{i=1}^{m} b_i. \tag{11}$$

Where a_j is the revenue when j is completed before the deadline, b_i is the compensation to users when the completion time of i is more than the deadline of i and m is the number of jobs completed after the deadline ($b_i < 0$).

5.2 Experiment Results

In Fig. 2, the resource utilization rate is not affected by the job set size in three algorithms. The resource utilization of TPS is the highest in the three algorithms and the resource utilization of EDF is slightly lower than that of TPS. The resource utilization of FIFO is the lowest. As shown in Fig. 3, due to the limited computing capacity, the job completion rates are reduced in three algorithms when the job set size

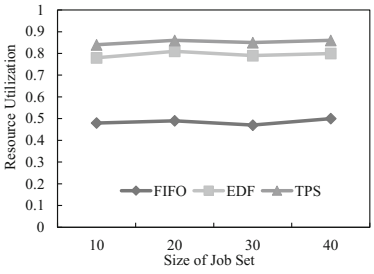

Fig. 2. The effect of job set size on *PRU*

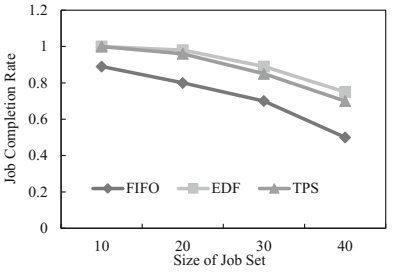

Fig. 3. The effect of job set size on *JCR*

increases. Because EDF and TPS consider the deadlines and TPS also takes into account the revenues, EDF has a higher completion rate than TPS. FIFO only considers the revenues so the algorithm has the lowest completion rate.

From Fig. 4, the revenues show a tendency to increase first and then decrease when the size of the job set increases for three algorithms. However, when the number of jobs exceeds the computing capacity of the platform, the number of jobs completed on time is reduced so that the total revenue declines. From Fig. 5, it can be seen that the wastage rates of the three algorithms are roughly same and decrease with the increase of the number of computing resources. However, the resource utilization of EDF and TPS increase with the increase of the number of computing resources and the TPS rise is more than that of EDF.

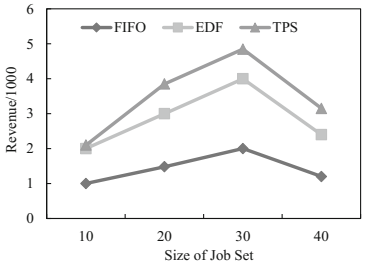

Fig. 4. The effect of job set size on *PR*

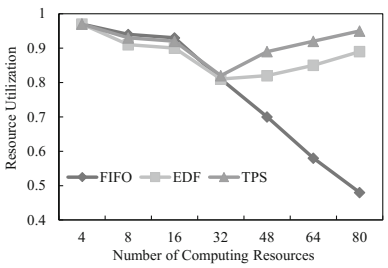

Fig. 5. The effect of resources on *PRU*

Figures 6 and 7 show the effect of the number of computing resources on job completion rate and the total revenue. It can be seen from the figures that with the increase of the number of computing resources, the completion rates and the total revenues are increased by TPS and EDF and the revenue of TPS is much larger than that of EDF. The job completion rate and the total revenue increase with the increase of resources in FIFO when the resources of the platform are less. However, because the jobs are executed serially to waste a large amount of resources, the job completion rate and the revenue do not increase with the increase of the resources and are stabilized at a fixed value when the resources of the platform are larger than the average job input size.

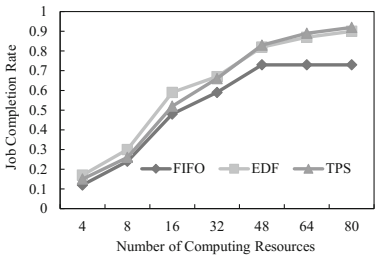

Fig. 6. The effect of job set size on *JCR*

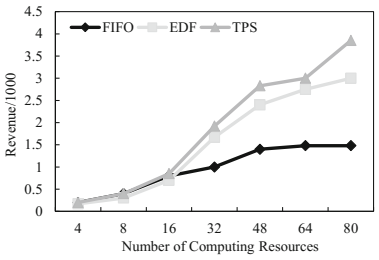

Fig. 7. The effect of resources on *PR*

6 Conclusion

The big data platform could serve multiple users at the same time. When users submitted jobs to the big data platform, jobs were reasonably scheduled not only to meet the requirements of users, but also improve the performance of the platform. Therefore, the two-stage job scheduling model was proposed for the jobs with the deadline constraints. In the model, the SMR algorithm calculated and adjusted the latest start time for every jobs based on the deadlines and the revenues of the jobs, which pre-allocated resources to jobs according to the result of adjustment to guarantee the jobs with the larger revenues to be completed before the deadlines. Under the premise of ensuring the maximum revenue, the SAS algorithm was developed to improve the resource utilization of the platform. Experimental results showed that the two-stage job scheduling model not only realized the maximum revenue of the provider, but improved the resource utilization of the platform. Moreover, the comprehensive performance of the platform was promoted.

Acknowledgments. The research work is supported by the TaiShan Industrial Experts Programme of Shandong Province No. tscy20150305, and the Key Research \& Development Program of Shandong Province No. 2016GGX101008, 2016ZDJS01A09.

References

1. Zhang, Z., Cherkasova, L., Verma, A., Loo, B.T.: Optimizing completion time and resource provisioning of pig programs. In: IEEEACM International Symposium on Cluster, Cloud and Grid Computing, pp. 811–816 (2012)
2. Khan, M., Jin, Y., Li, M., Xiang, Y., Jiang, C.J.: Hadoop performance modeling for job estimation and resource provisioning. Parallel Distrib. Syst. **27**(2), 441–454 (2016)
3. Verma, A., Cherkasova, L., Campbell, R.H.: ARIA: automatic resource inference and allocation for mapreduce environments. In: International Conference on Autonomic Computing, pp. 235–244 (2011)
4. Cheng, D., Rao, J., Jiang, C., Zhou, X.: Resource and deadline-aware job scheduling in dynamic hadoop clusters. In: Parallel and Distributed Processing Symposium, pp. 956–965 (2015)
5. Li, S., Hu, S., Wang, S., Su, L., Abdelzaher, T., Gupta, I., Pace, R.: WOHA: deadline-aware map-reduce workflow scheduling framework over hadoop clusters. In: Distributed Computing Systems, pp. 93–103 (2014)
6. Liu, X., Wang, C., Zhou, B.B., Chen, J., Yang, T., Zomaya, A.Y.: Priority-based consolidation of parallel workloads in the cloud. Parallel Distrib. Syst. **24**(9), 1874–1883 (2013)
7. Koutsandria, G., Skevakis, E., Sayegh, A.A.: Can everybody be happy in the cloud? Delay, profit and energy-efficient scheduling for cloud services. J. Parallel Distrib. Comput. **96**, 202–217 (2016)
8. Clinkenbeard, T., Nica, A.: Job Scheduling with minimizing data communication costs. In: ACM International Conference on Management of Data, pp. 2071–2072 (2015)
9. Wang, Q., Li, X., Wang, J.: A data placement and task scheduling algorithm in cloud computing. J. Comput. Res. Dev. **51**(11), 2416–2426 (2014)

10. Sun, M., Zhuang, H., Li, C., Lu, K.: Scheduling algorithm based on prefetching in mapreduce clusters. Appl. Soft Comput. **38**(C), 1109–1118 (2016)
11. Cao, J., Zeng, G., Niu, J., Xu, J.: Availability-aware scheduling method for parallel task in cloud environment. J. Comput. Res. Dev. **50**(7), 1563–1572 (2013)
12. Zheng, X., Xiang, M., Zhang, D., Liu, Q.: An adaptive tasks scheduling method based on the ability of node in hadoop cluster. J. Comput. Res. Dev. **51**(3), 618–626 (2014)
13. Li, Z., Chen, S., Yang, B., Li, R.: Multi-objective memetic algorithm for task scheduling on heterogeneous cloud. Chin. J. Comput. **39**(2), 377–390 (2016)
14. Lee, M.C., Lin, J.C., Yahyapour, R.: Hybrid job-driven scheduling for virtual mapreduce clusters. Parallel Distrib. Syst. **27**(6), 1687–1699 (2016)
15. Wang, X., Shen, D., Bai, M., Nie, T., Kou, Y., Yu, G.: Sames: deadline-constraint scheduling in mapreduce. Front. Comput. Sci. **9**(1), 128–141 (2015)
16. Wang, Y., Shi, W.: Budget-driven scheduling algorithms for batches of mapreduce jobs in heterogeneous clouds. Cloud Comput. **2**(3), 306–319 (2014)
17. Song, Y., Sun, Y., Shi, W.: A two-tiered on-demand resource allocation mechanism for VM-based data centers. Serv. Comput. **6**(1), 116–129 (2013)
18. Liu, W., Wang, Z., Shen, Y.: Job-aware network scheduling for hadoop cluster. KSII Trans. Internet Inf. Syst. **11**(1), 237–252 (2017)
19. Kumar, K.A., Konishetty, V.K., Voruganti, K., Rao, G.V.P.: Cash: context aware scheduler for hadoop. In: International Conference on Advances in Computing, Communications and Informatics, pp. 52–61 (2012)
20. Mao, M., Humphrey, M.: Auto-scaling to minimize cost and meet application deadlines in cloud workflows. In: High PERFORMANCE Computing, Networking, Storage and Analysis, pp. 1–12 (2011)
21. Gasior, J., Seredynski, F.: Metaheuristic approaches to multiobjective job scheduling in cloud computing systems. In: IEEE International Conference on Cloud Computing Technology and Science, pp. 222–229 (2016)
22. Wang, X., Shen, D., Yu, G., Bai, M., Nie, T., Kou, Y.: Resource on maximum benefit problem in a mapreduce cluster. Chin. J. Comput. **38**(1), 109–121 (2015)