J Comb Optim (2018) 35:216–223
https://doi.org/10.1007/s10878-017-0167-4

Online MapReduce processing on two identical parallel machines

Jidan Huang[1] · **Feifeng Zheng**[1] · **Yinfeng Xu**[1] · **Ming Liu**[2]

Published online: 20 August 2017

Abstract In this work we investigate the online over-list MapReduce processing problem on two identical parallel machines, aiming at minimizing the makespan. Jobs are revealed one by one, and each job consists of one map task and one reduce task. The map task can be arbitrarily split and processed on both machines simultaneously, while the reduce task has to be processed on a single machine and it cannot be started unless the map task has been completed. We first show that the general case of the problem reduces to the classical two machine online scheduling model with an optimal competitive ratio of 3/2. For a special case where the map task is at least as long as the reduce task, we prove that no online algorithm can be less than 4/3-competitive. An optimal Greedy algorithm with a matching competitive ratio is proposed as well.

Keywords MapReduce · Online scheduling · Competitive ratio

✉ Ming Liu
mingliu@tongji.edu.cn

Jidan Huang
haungjd@dhu.edu.cn

Feifeng Zheng
ffzheng@dhu.edu.cn

Yinfeng Xu
xyf@dhu.edu.cn

[1] Glorious Sun School of Business and Management, Donghua University, Shanghai 200051, China

[2] School of Economics and Management, Tongji University, Shanghai 200092, China

1 Introduction

In the last decade, the MapReduce system under the big data processing framework has caught much interest in operations research (Dean and Ghemawat 2008; Sandholm and Lai 2009; Chen et al. 2012). MapReduce processing generally consists of two phases: the *map* phase and the *reduce* phase. Any job in the system consists of one map task and one reduce task. The map task can be arbitrarily split and processed simultaneously on parallel machines, while the reduce task of the job can only be processed on one of the machines and it starts after the completion of the map task.

Moseley et al. (2011) modeled the MapReduce system as the two-stage flexible flow shop problem. For the objective of the total flow time they presented a 12-approximation algorithm. They also designed a dynamic program for the case where jobs are of the same release time. Zhu et al. (2014) investigated the MapReduce problems of minimizing either makespan or total completion time. They considered both preemptive and non-preemptive reduce tasks. For the objective of makespan minimization, they proved that the preemptive model is solvable, and proposed a $(\frac{3}{2} - \frac{1}{2m})$-approximation algorithm for the non-preemptive model where m is the number of machines. For the objective of total completion time, they presented a $2 - 1/m$-approximation algorithm for the non-preemptive model and a heuristic for the preemptive model.

Some authors paid attention to the online version of the MapReduce processing problem, and constructed competitive online algorithms for various online models. Zheng et al. (2013) investigated the online over-time problem of minimizing the total flow time of jobs in the MapReduce framework. They showed that for non-preemptive tasks, no online algorithms guarantee constant competitive ratios. They constructed a slightly weaker metric of performance called the efficiency ratio, and presented an online algorithm with a small efficiency ratio. Chen et al. (2017) considered the online over-time MapReduce systems where jobs are released over time and the objective is to minimize the makespan. The map task of any job is parallelized while the reduce task of the job can only be processed on one of the machines. For non-preemptive model, they proposed a $2 - 1/m$-competitive algorithm called Map First-Longest Processing Time (MF-LPT), and a lower bound is provided as well. For preemptive-resumption model, they proposed an optimally 1-competitive algorithm for the case with two parallel machines.

Luo et al. (2017) studied the online over-list MapReduce processing such that jobs are revealed one-by-one. They assumed that the reduce task of any job is unknown until its map task has been completed. For the makespan minimization objective, they proved a lower bound of $2 - 1/m$ for any deterministic online algorithm where m is the number of parallel machines. For both preemptive-resumption and non-preemptive models, they proposed optimally $2 - 1/m$-competitive online algorithms.

In this work, we investigate the online over-list model and focus on the two parallel machine environment. Different from the model in Luo et al. (2017), we assume that both the map and the reduce tasks of a job become known on the release of the job. We first show that for the general case with no constraints on the processing times of map tasks and reduce tasks, the problem reduces to the classical two parallel machine

scheduling problem. For a special case with map tasks longer than reduce tasks, we propose an optimally 4/3-competitive algorithm.

2 Problem description

There are two parallel identical machines in the online MapReduce processing system to process jobs which are revealed one-by-one over list. Each job consists of one *Map* task and one *Reduce* task, and the processing times of both tasks become known to the decision maker once the job is revealed. A decision on the processing schedule of both map and reduce tasks has to be made immediately and irrecoverably on the release of the job. Moreover, the map task is required to be completed on or before the start of the reduce task. Preemption for the processing of either map or reduce task is not allowed. In the MapReduce system, each map task is parallelizable, i.e., it can be arbitrarily split and processed on the two machines simultaneously, while the reduce task can only be processed on a single machine. The objective of the problem is to minimize the makespan, i.e., the largest completion time of job.

We represent any job j as (M_j, R_j) where M_j and R_j are the processing of the map task and respectively the reduce task. Denote by $|M_j|$ and $|R_j|$ the respective processing time of the corresponding task. Given an arbitrary processing schedule produced by an online algorithm, let $S(M_j)$ (or $S(R_j)$) the start time of the map task (or the reduce task) of job j, and $C(M_j)$ (or $C(R_j)$) the end time of the map task (or the reduce task) of the job. Let $C_{\max}$ and $C^*_{\max}$ be the objective value of the schedule by the online algorithm and that of an optimal schedule, respectively. Adopting the classical three-fold notation, we denote the above problem as $P2|online, over - list, MapReduce|C_{\max}$.

The performance of any online algorithm $\mathcal{A}$ is usually measured by *competitive ratio* (Fiat and Woeginger 1998). Consider an online problem with a minimization objective, we say $\mathcal{A}$ is ρ-competitive if for any job input instance Γ, $C_{\mathcal{A}}(\Gamma) \leq \rho C^*(\Gamma) + b$ where $b \geq 0$ is any real constant, and $C_{\mathcal{A}}(\Gamma)$ and $C^*(\Gamma)$ are the minimization objective value of the schedule by $\mathcal{A}$ and that of an optimal schedule, respectively. We also say $\mathcal{A}$ has a *competitive ratio* of ρ' which is the infimum of ρ satisfying the above inequality.

3 The general case of $P2|online, over - list, MapReduce|C_{\max}$

In Luo et al. (2017), it is proved for the general case with no constraints for $|M_j|$ and $|R_j|$ that any online algorithm has a competitive ratio at least $2 - 1/m$ where m is the number of machines. It means a lower bound of 3/2 for the two machine case. They presumed that the value of $|R_j|$ is unknown until the map task M_j has been completed.

In this section, we consider the general case for problem $P2|online, over - list, MapReduce|C_{\max}$, in which both $|M_j|$ and $|R_j|$ become known on the release of job j. We first prove a lower bound of 3/2, and then point out that the considered problem reduces to the classical two machine online scheduling model.

Theorem 1 *For problem* $P2|online, over - list, MapReduce|C_{\max}$, *any online algorithm cannot have a competitive ratio less than* 3/2.

Proof To prove the theorem, it suffices to construct a specific job input instance Γ such that any online algorithm $\mathcal{A}$ cannot produce a processing schedule with makespan $C_{\max}^{\mathcal{A}}$ less than 3/2 times of the makespan $C_{\max}^{*}$ of an optimal schedule. Γ consists of at most three jobs.

The first two jobs 1 and 2 are of processing time $(|M_j|, |R_j|) = (0, 1)$ for $j = 1, 2$. Consider the following two cases by the behavior of $\mathcal{A}$.

Case 1 Both jobs are assigned to the same machine. In this case, no more jobs arrive. $C_{\max}^{\mathcal{A}} = \max\{C(R_1), C(R_2)\} \geq |R_1| + |R_2| = 2$. In an optimal schedule the two jobs are processed on two machines with $C_{\max}^{*} = |R_1| = 1$. The ratio $\frac{C_{\max}^{\mathcal{A}}}{C_{\max}^{*}} \geq 2$ in this case.

Case 2 R_1 and R_2 are assigned to different machines. In this case, the third and last job 3 with $(|M_3|, |R_3|) = (0, 2)$ is released. The assignment of the reduce task R_3 of job 3 is the same as either R_1 or R_2, it implies that $C_{\max}^{\mathcal{A}} \geq 2 + 1 = 3$. In an optimal schedule, however, jobs 1 and 2 are assigned to one machine and job 3 is assigned to the other machine. $C_{\max}^{*} = |R_3| = 2$, and thus $\frac{C_{\max}^{\mathcal{A}}}{C_{\max}^{*}} \geq \frac{3}{2}$ in this case.

Hence, $\frac{C_{\max}^{\mathcal{A}}}{C_{\max}^{*}} \geq \frac{3}{2}$ for any case. The theorem is established. □

It is well known that for the classical $P2|online, over - list|C_{\max}$ problem, any online algorithm cannot be better than 3/2-competitive in worst-case analysis (Faigle et al. 1989), and LIST algorithm is optimally 3/2-competitive (Graham 1966). We observe that in the above proof of Theorem 1, all the three jobs in the instance are of reduce tasks but not map tasks. As any reduce task can only be processed by a single machine, we have the following conclusion for the problem $P2|online, over - list, MapReduce|C_{\max}$.

Theorem 2 *Problem* $P2|online, over-list, MapReduce|C_{\max}$ *reduces to the classical* $P2|online, over - list|C_{\max}$ *problem in worst-case analysis.*

We may explain the above theorem as follows. For $P2|online, over - list, MapReduce|C_{\max}$, the map task M_j of any job j can be arbitrarily split and processed on the two machines simultaneously. If $|M_j| > 0$ for some job j, then an online algorithm $\mathcal{A}$ has more flexibility to schedule job j, compared with the case with $|M_j| = 0$. The optimal schedule, however, always makes the two machines be of the same workload in worst-case analysis, no matter whether $|M_j| = 0$ or not. The case with $|M_j| > 0$ results in an improvement of the competitive performance of $\mathcal{A}$. Thus, in worst-case analysis, $|M_j| = 0$ for each job j, implying the above theorem.

4 A special case with map tasks longer than reduce tasks

We observe that Theorem 2 also applies to the special case where $|M_j| < |R_j|$ for any job j, implying that no online algorithm can be better than 3/2-competitive. In what follows, we focus on the other case where $|M_j| \geq |R_j|$ for any job j. We first give a lower bound of competitive ratio for the case, and then present an online algorithm and prove its optimal competitiveness.

4.1 A lower bound of competitive ratio

Theorem 3 *In problem* $P2|online, over-list, MapReduce|C_{\max}$, *if* $|M_j| \geq |R_j|$ *for any job* j, *then any online algorithm cannot have a competitive ratio less than* $4/3$.

Proof To prove the theorem, it suffices to construct a job input instance Γ to make the makespan of the processing schedule produced by any online algorithm $\mathcal{A}$ is least $4/3$ times of that of an optimal schedule. Γ consists of at least two jobs. The first two jobs 1 and 2 are of identical processing time, i.e., $(|M_j|, |R_j|) = (1, 1)$ for $j = 1, 2$. Assume without loss of generality that machine 1 is of a workload not less than machine 2 after the assignment of the first two jobs. Consider the following two cases.

Case 1 Machine 1 is of a workload at least $8/3$ while machine 2 has a workload at most $4/3$ for processing the first two jobs. In this case, no more jobs are released. $C^{\mathcal{A}}_{\max} \geq 8/3$. In an optimal schedule, however, the two jobs are processed on different machines with $C^*_{\max} = |M_1| + |R_1| = 2$. Thus, the ratio of $\frac{C^{\mathcal{A}}_{\max}}{C^*_{\max}} \geq \frac{4}{3}$ in this case.

Case 2 Machine 2 is of a workload at least $4/3$, and the two jobs are completed on or before time $8/3$ on machine 1. In this case, the third and last job 3 with $(|M_3|, |R_3|) = (4, 4)$ is released. No matter how $\mathcal{A}$ schedules job 3, its reduce task R_3 is started not earlier than $\frac{\sum_{j=1}^{2}(|M_j|,|R_j|)+|M_3|}{2} = 4$, and $C^{\mathcal{A}}_{\max} \geq S(R_3) + |R_3| \geq 4 + 4 = 8$. In an optimal schedule, the map task of job 3 is processed on both machines during $[0, 2)$, and then the first two jobs are processed on one machine from time 2, and the reduce task of job 3 is processed on the other machine. $C^*_{\max} = \frac{|M_3|}{2} + |R_3| = 2 + 4 = 6$. Again, the ratio of $\frac{C^{\mathcal{A}}_{\max}}{C^*_{\max}} \geq \frac{4}{3}$ in this case.

Hence, $\frac{C^{\mathcal{A}}_{\max}}{C^*_{\max}} \geq \frac{4}{3}$ in the above both cases. The theorem is established. □

Figure 1 illustrates the online and the optimal schedules for the two cases in the proof of Theorem 1. In the figure, $Mac1$ and $Mac2$ denote machines 1 and 2, respectively. Figure 1a, c denote an online schedule and respectively an optimal schedule in Case 1 of the proof, while Fig. 1b, d are for that in Case 2.

4.2 An optimal online algorithm

Below we present an online algorithm named Greedy, which always starts the reduce task of each job as soon as possible. We then prove that it behaves optimally with a matching competitive ratio of $4/3$. Remember that the map task of any job can be partitioned into more than one piece and processed simultaneously on the two machines. We formally describe the online algorithm as follows.

Algorithm Greedy
On the release of job j ($j \geq 1$), let t_1 and t_2 be the current completion times of machines 1 and 2, respectively. Assume without loss of generality that $t_1 \leq t_2$. Notice that machines 1 and 2 are kept busy in processing jobs during time periods $[0, t_1)$ and $[0, t_2)$, respectively. The Greedy starts the map task M_j of job j at

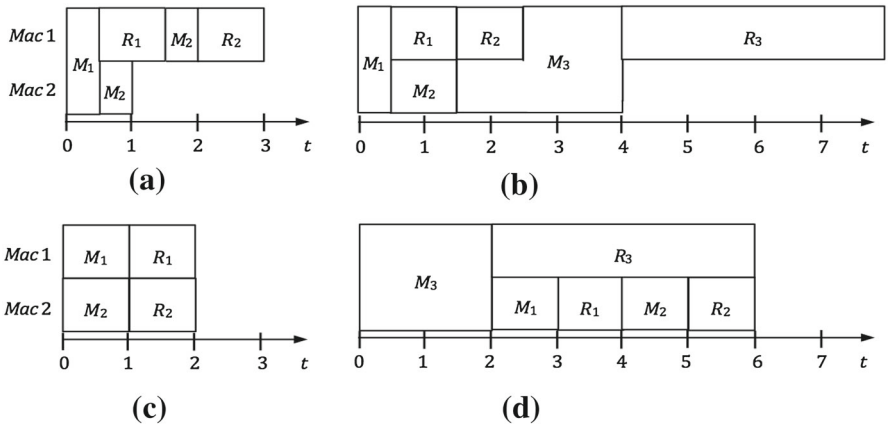

Fig. 1 Illustration of the proof of Theorem 3. **a** An online schedule in Case 1, **b** an online schedule in Case 2, **c** an optimal schedule in Case 1, **d** an optimal schedule in Case 2

time t_1 on machine 1. **Case (1)** $t_2 - t_1 \geq |M_j|$. In this case, the reduce task R_j is started at time $S(R_j) = C(M_j) = t_1 + |M_j|$, and it is completed at time $C(R_j) = t_1 + |M_j| + |R_j|$.

Case (2) $t_2 - t_1 < |M_j|$. Then $S(R_j) = C(M_j) = t_2 + \frac{|M_j| - (t_2 - t_1)}{2} = \frac{t_1 + t_2 + |M_j|}{2}$. That is, the remaining workload, i.e., $|M_j| - (t_2 - t_1)$, of task M_j at time t_2 is partitioned into two equal segments and assigned to both machines. R_j is then assigned to machine 1 on the completion of M_j, and $C(R_j) = S(R_j) + |R_j|$.

We observe that if Case (2) of the online algorithm occurs for some job j, then $S(R_j) = \frac{\sum_{u=1}^{j-1}(|M_u| + |R_u|) + |M_j|}{2}$. That is, the two machines are of the same workload on the start of task R_j. In the following we prove the optimal competitiveness of the Greedy algorithm.

Theorem 4 *For problem* $P2|online, over - list, MapReduce|C_{\max}$, *given that* $|M_j| \geq |R_j|$ *for any job* j, *Greedy is optimally* $4/3$*-competitive.*

Proof On the release of any job j, assume that machines 1 and 2 complete their current workloads at times t_1 and t_2 respectively. Note that

$$t_1 + t_2 = \sum_{u=1}^{j-1}(|M_u| + |R_u|). \tag{1}$$

We assume without loss of generality that $t_1 < t_2$. For the other case with $t_1 \geq t_2$, it can be similarly analyzed.

We first bound the value of t_1. As $t_1 < t_2$, the reduce task R_{j-1} of the preceding job $j-1$ is processed on machine 2, $S(R_{j-1}) = C(M_{j-1}) = S(M_{j-1}) + |M_{j-1}|$, and $C(R_{j-1}) = S(R_{j-1}) + |R_{j-1}| = t_2$ by the algorithm. If $t_1 > C(M_{j-1}) = S(R_{j-1})$, then $t_1 > \frac{\sum_{u=1}^{j-2}(|M_u| + |R_u|) + |M_{j-1}|}{2} \geq \frac{\sum_{u=1}^{j-1}|M_u|}{2}$. Otherwise $t_1 = C(M_{j-1}) =$

$\frac{\sum_{u=1}^{j-2}(|M_u|+|R_u|)+|M_{j-1}|}{2} \geq \frac{\sum_{u=1}^{j-1}|M_u|}{2}$ where the above first equation is by Case (2) of the algorithm. Since $|R_u| \leq |M_u|$ for $1 \leq u \leq j-1$, we conclude that $t_1 \geq \frac{\sum_{u=1}^{j-1}|M_u|}{2} \geq \frac{\sum_{u=1}^{j-1}(|M_u|+|R_u|)/2}{2} = \frac{t_1+t_2}{4}$ where the last equation is by Eq. (1). The above inequality is reformulated as

$$t_1 \geq \frac{t_2}{3}. \tag{2}$$

Now we consider the following two cases depending on the values of t_1 and t_2 on the release of any job j (≥ 1).

Case 1 $t_2 - t_1 \geq |M_j|$, i.e., Case (1) of the Greedy algorithm happens. Then job j is processed on machine 1, and it is completed at time $C(R_j) = t_1 + |M_j| + |R_j|$. In an optimal schedule, assume without loss of generality that j is the last completed job. then $C^*(R_j) \geq \frac{t_1+t_2+|M_j|+|R_j|}{2}$.

$$\begin{aligned}\frac{C(R_j)}{C^*(R_j)} &\leq \frac{2(t_1+|M_j|+|R_j|)}{t_1+t_2+|M_j|+|R_j|}\\ &\leq \frac{2(t_1+2|M_j|)}{t_1+t_2+2|M_j|}\\ &\leq \frac{4t_2-2t_1}{3t_2-t_1}\\ &\leq \frac{5}{4}\end{aligned}$$

where the above second inequality is due to $|M_j| \geq |R_j|$, and the third inequality is by the case condition $|M_j| \leq t_2 - t_1$. The fourth inequality is by Inequality (2).

Case 2 $t_2 - t_1 < |M_j|$. By Case (2) of the algorithm, $S(R_j) = C(M_j) = \frac{t_1+t_2+|M_j|}{2}$, and $C(R_j) = S(R_j)+|R_j| = \frac{t_1+t_2+|M_j|+2|R_j|}{2}$ in this case. In an optimal schedule, we have $C^*(R_j) \geq \frac{t_1+t_2+|M_j|+|R_j|}{2} = \frac{\sum_{u=1}^{j}(|M_u|+|R_u|)}{2}$ by Eq. (1). Moreover, $C^*(R_j) \geq \frac{|M_j|}{2} + |R_j|$ since task R_j can only be processed on a single machine and it is started on or after the completion of M_j. If $t_1 + t_2 \geq |R_j|$, then $C^*(R_j) \geq \frac{t_1+t_2+|M_j|+|R_j|}{2}$, and

$$\begin{aligned}\frac{C(R_j)}{C^*(R_j)} &\leq 1 + \frac{|R_j|/2}{(t_1+t_2+|M_j|+|R_j|)/2}\\ &\leq 1 + \frac{|R_j|}{|M_j|+2|R_j|}\\ &\leq \frac{4}{3}\end{aligned}$$

where the second inequality is due to $t_1 + t_2 \geq |R_j|$, and the third inequality is by $|R_j| \leq |M_j|$. Otherwise if $t_1 + t_2 < |R_j|$, then $C^*(R_j) \geq \frac{|M_j|}{2} + |R_j|$, and

$$\begin{aligned}\frac{C(R_j)}{C^*(R_j)} &\leq \frac{(t_1 + t_2 + |M_j| + 2|R_j|)/2}{|M_j|/2 + |R_j|} \\ &< \frac{(|M_j| + 3|R_j|)/2}{|M_j|/2 + |R_j|} \\ &\leq \frac{4}{3}\end{aligned}$$

where the second and third inequalities are due to $t_1 + t_2 < |R_j|$ and $|R_j| \leq |M_j|$, respectively.

According to the above two cases, after the assignment of any job j, the current makespan of the schedule produced by the Greedy is at most $4/3$ times of that of an optimal solution. It implies that Greedy is $4/3$-competitive. The theorem is established. □

5 Conclusions

This work investigates the online MapReduce problem on two parallel machines with the objective of makespan minimization. We consider two cases where the map task of each job is shorter than the reduce task and where the map task is longer than the reduce task. For the former case we reveal that it reduces to the classical online scheduling on two parallel machines. For the latter case, we prove a lower bound of $4/3$, and put forward an online algorithm with a matching competitive ratio. One further research is to extend the above results to the environment with $m \geq 2$ parallel machines.

References

Chen F, Kodialam M, Lakshman TV (2012) Joint scheduling of processing and shuffle phases in MapReduce systems. In: INFOCOM, 2012 Proceedings IEEE, pp 1143–1151

Chen C, Xu Y, Zhu Y, Sun C (2017) Online MapReduce scheduling problem of minimizing the makespan. J Comb Optim 33:590–608

Dean J, Ghemawat S (2008) MapReduce: simplified data processing on large clusters. Commun ACM 51(1):107–113

Faigle U, Kern W, Turan G (1989) On the performance of on-line algorithms for partition problems. Acta Cybern 9:107–119

Fiat A, Woeginger G (1998) Competitive analysis of algorithms. In: LNCS 1442. Springer Berlin, pp 1–12

Graham RL (1966) Bounds for certain multiprocessor anomalies. Bell Syst Tech J 45:1563–1581

Luo T, Zhu Y, Wu W, Xu Y, Du D (2017) Online makespan minimization in MapReduce-like systems with complex reduce tasks. Optim Lett 11:271–277

Moseley B, Dasgupta A, Kumar R, Sarlós T (2011) On scheduling in map-reduce and flow-shops. In: Proceedings of the twenty-third annual ACM symposium on parallelism in algorithms and architectures, ACM, SPAA, vol 11, pp 289–298

Sandholm T, Lai K (2009) MapReduce optimization using regulated dynamic prioritization. SIGMETRICS Perform Eval Rev 37(1):299–310

Zheng Y, Shroff N, Sinha P (2013) A new analytical technique for designing provably efficient MapReduce schedulers. In: INFOCOM, 2013 Proceedings IEEE, pp 1600–1608

Zhu Y, Jiang Y, Wu W, Ding L, Teredesai A, Li D, Lee W (2014) Minimizing makespan and total completion time in MapReduce-like systems. In: INFOCOM, 2014 Proceedings IEEE, pp 2166–2174