Optimization Letters (2019) 13:1663–1676
https://doi.org/10.1007/s11590-018-01384-8

ORIGINAL PAPER

Optimal online algorithms for MapReduce scheduling on two uniform machines

Yiwei Jiang[1] · Ping Zhou[2] · T. C. E. Cheng[3] · Min Ji[1]

Received: 20 July 2018 / Accepted: 29 December 2018 / Published online: 11 January 2019

Abstract
We study online scheduling on two uniform machines in the MapReduce system. Each job consists of two sets of tasks, namely the map tasks and reduce tasks. A job's reduce tasks can only be processed after all its map tasks are finished. The map tasks are fractional, i.e., they can be arbitrarily split and processed on different machines in parallel. Our goal is to find a schedule that minimizes the makespan. We consider two variants of the problem, namely the cases involving preemptive reduce tasks and non-preemptive reduce tasks. We provide lower bounds for both variants. For preemptive reduce tasks, we present an optimal online algorithm with a competitive ratio of $\frac{\sqrt{s^2+2s+5}+1-s}{2}$, where $s \geq 1$ is the ratio between the speeds of the two machines. For non-preemptive reduce tasks, we show that the LS-like algorithm is optimal and its competitive ratio is $\frac{2s+1}{s+1}$ if $s < \frac{1+\sqrt{5}}{2}$ and $\frac{s+1}{s}$ if $s \geq \frac{1+\sqrt{5}}{2}$.

Keywords Big data · MapReduce scheduling · Online algorithm · Competitive ratio

1 Introduction

In this paper we consider the problem of online scheduling on two uniform machines in the MapReduce system [3], which is a programming model and the associated implementation for processing large-scale data. It is a two-phase paradigm consisting of the map phase and the reduce phase. Specifically, when a job is submitted for processing by the system, the system partitions it into two types of tasks, namely the map tasks and reduce tasks. The map tasks take the raw data as input and output

✉ Min Ji
jimkeen@163.com

1 School of Management and E-Business, Contemporary Business and Trade Research Center, Zhejiang Gongshang University, Hangzhou 310018, China

2 College of Humanities, Zhejiang Business College, Hangzhou 310053, China

3 Department of Logistics and Maritime Studies, The Hong Kong Polytechnic University, Kowloon, Hong Kong

the key-value pairs, while the reduce tasks take the pairs output by the map tasks and compute the final results. All the map tasks and reduce tasks are processed on a number of parallel processors.

Formally, we introduce the problem under study as follows. A sequence $\mathcal{J}$ of independent jobs $J_1, J_2, \ldots, J_n$ arrive one by one that are to be scheduled irrevocably onto two uniform machines σ_1 and σ_2. Machine σ_i has speed $s_i \geq 1$. Without loss of generality, we assume $s = s_1 \geq s_2 = 1$. Each job J_j has a set of map tasks $M_j = \{m_j^1, m_j^2, \ldots, m_j^{u_j}\}$ and a set of reduce tasks $R_j = \{r_j^1, r_j^2, \ldots, r_j^{v_j}\}$, i.e., the job has u_j map tasks and v_j reduce tasks. The reduce tasks of J_j can only be processed after all the map tasks of J_j are finished. We assume that the map tasks are *fractional*, i.e., each map task can be arbitrarily split and parts of the same task can be processed on different machines in parallel. Our goal is to minimize the makespan, i.e., the maximum completion time of all the jobs. We consider two variants of the problem, namely the cases involving preemptive reduce tasks and non-preemptive reduce tasks. In the former case, each reduce task may be cut into a few pieces, which are assigned to possibly different machines for processing in non-overlapping time slots, while in the latter case, each reduce task must be processed on one machine in a continuous time interval. Using the three-field notation for scheduling problems, we denote our problem as $Q2|M(\mathit{frac})R(\mathit{pmtn})|C_{\max}$ for the case involving preemptive reduce tasks and $Q2|M(\mathit{frac})R|C_{\max}$ for the case involving non-preemptive reduce tasks.

The performance of an online algorithm A is measured by its competitive ratio, which is defined as $\rho_A = \inf\{\rho \geq 1 | C^A(\mathcal{J}) \leq \rho \cdot C^*(\mathcal{J}), \forall \mathcal{J}\}$, where $C^A(\mathcal{J})$ (or in short C^A) denotes the objective value produced by A and $C^*(\mathcal{J})$ (or in short C^*) denotes the optimal value in the offline version. An online problem has a *lower bound* if no online algorithm has a competitive ratio smaller than the lower bound. An online algorithm is called *optimal* if its competitive ratio matches its largest lower bound.

For MapReduce scheduling on parallel machines to minimize the makespan, Zhu et al. [13] considered offline scheduling on m identical machines where the map tasks are fractional, i.e., $Pm|M(\mathit{frac}), \mathit{offline}|C_{\max}$. They presented an algorithm that produces an optimal solution for preemptive reduce tasks and an algorithm with a worst-case ratio of $\frac{3}{2} - \frac{1}{2m}$ for non-preemptive tasks. They also considered the problem to minimize the total completion time, and gave an approximation algorithm and a heuristic for the non-preemptive and preemptive cases, respectively. For the offline version of the problem on m uniform machines, i.e., $Qm|M(\mathit{frac}), \mathit{offline}|C_{\max}$, Jiang et al. [6] provided an approximation algorithm with a worst-case ratio of 2 for the preemptive case and an approximation algorithm with a worst-case ratio of $\max\{1 + \frac{\Delta}{2} - \frac{1}{m}, \Delta\}$, where Δ is the speed ratio between the fastest and slowest machines, for the non-preemptive case.

For the online version of the problem in [13], Jiang et al. [7] considered preemptive scheduling on two machines, i.e., $P2|M(\mathit{frac})R(\mathit{pmtn}), \mathit{online}|C_{\max}$. They presented an optimal online algorithm with a competitive ratio of $\sqrt{2}$. Luo et al. [8] considered the online version under the assumption that a job's reduce tasks are unknown until its map tasks are finished. They provided online optimal algorithms with the same competitive ratio of $2 - \frac{1}{m}$ for both the preemptive and non-preemptive cases. Huang et al. [5] studied a special case of online over-list MapReduce model on two identical machines where each job consists of only one map task and one reduce task. When jobs are released over time, Chen et al. [2] presented an algorithm with a competitive ratio $2 - \frac{1}{m}$ for the non-preemptive case and an optimal algorithm for the preemptive case involving two machines. Besides, some researchers studied different online MapReduce scheduling models with different objectives. Le et al. [9] considered online MapReduce load balancing with skewed data input. They presented a 2-competitive ratio algorithm and a sample-based enhancement that probabilistically achieves a 3/2-competitive ratio with a bounded error. Zheng et al. [12] introduced the criterion of efficiency ratio to measure the performance of online algorithms for the problem to minimize the total completion time. Under the assumption that the length of each map task is one unit, they provided an algorithm based on the shortest remaining processing time (*SRPT*) and showed that it has a small efficiency ratio for the preemptive case. Chang et al. [1] considered the MapReduce scheduling to minimize the weighted total completion time and presented an online algorithm that achieves a 30% improvement over FIFO via simulation. Moseley et al. [10] considered the problem to minimize the total flowtime in the two-stage hybrid flow shop where the speed of each machine is $1+\epsilon, 0 < \epsilon \leq 1$. They presented an online $1+\epsilon$-speed $O(\frac{1}{\epsilon^2})$-competitive algorithm.

In this paper we consider online MapReduce scheduling on two uniform machines in which the map tasks are fractional. Though Jiang et al. [6] presented online heuristics for the problem and confirmed the advantage of the algorithms through simulation, they did not provide the competitive ratios of the online algorithms. Our contribution is to provide two optimal online algorithms for the cases involving preemptive and non-preemptive reduce tasks. For the variant $Q2|M(\mathit{frac})R(\mathit{pmtn}), \mathit{online}|C_{\max}$, we propose an optimal online algorithm with a competitive ratio of $\frac{\sqrt{s^2+2s+5}+1-s}{2}$, which is strictly greater than the competitive ratio $\frac{(s+1)^2}{s^2+s+1}$ for the classical preemptive uniform-machine scheduling problem $Q2|\mathit{pmtn}, \mathit{online}|C_{\max}$ [11], as shown in Fig. 1. Note that $\frac{\sqrt{s^2+2s+5}+1-s}{2} = \sqrt{2}$ when $s = 1$, so it is a general result for the problem in [7]. For the variant $Q2|M(\mathit{frac})R, \mathit{online}|C_{\max}$, we propose an optimal online algorithm with a competitive ratio of $\frac{2s+1}{s+1}$ if $s < \frac{1+\sqrt{5}}{2}$ and $\frac{s+1}{s}$ if $s \geq \frac{1+\sqrt{5}}{2}$, which is the same as that of the classical non-preemptive uniform-machine scheduling problem $Q2|\mathit{online}|C_{\max}$ [4].

We organize the rest of the paper as follows: in Sect. 2 we derive lower bounds for the problem. In Sects. 3 and 4 we present optimal online algorithms for the preemptive and non-preemptive variants, respectively. Finally, we conclude the paper in Sect. 5.

2 Lower bounds

For the sake of simplicity in the remainder of this paper, we use r_j^i to denote the length of a task, and M_j (R_j) to denote the total length of the tasks in M_j (R_j). We assume that $r_j^1 \geq r_j^2 \geq \cdots \geq r_j^{v_j}$ for every $1 \leq j \leq n$. Let $P_j = \sum_{i=1}^{j}(M_i + R_i)$ be the total length of the first j jobs. Let C_j^A and C_j^* denote the makespan produced by algorithm A and the optimal makespan after scheduling the first j jobs.

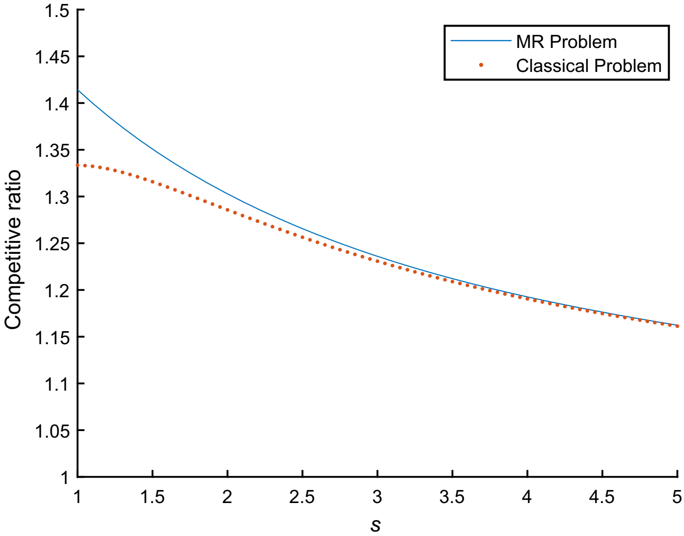


Fig. 1 Comparison between two competitive ratios

Lemma 2.1 *For both variants $Q2|M(frac)R(pmtn)$, $online|C_{\max}$ and $Q2|M(frac)R$, $online|C_{\max}$, we have*

$$C_j^* \geq \max\left\{\frac{P_j}{1+s}, \max_{1\leq i\leq j}\left\{\frac{r_i^1}{s}+\frac{M_i}{1+s}\right\}\right\}. \tag{1}$$

We first derive a lower bound for the preemptive variant $Q2|M(frac)R(pmtn)$, $online|C_{\max}$.

Theorem 2.2 *For the variant $Q2|M(frac)R(pmtn)$, $online|C_{\max}$, the competitive ratio of any online algorithm is at least $\frac{\sqrt{s^2+2s+5}+1-s}{2}$.*

Proof Assume that there exists an algorithm A with a competitive ratio ρ. The first job $J_1 = \{M_1, \emptyset\}$ arrives. Let l_i be the completion time of machine σ_i after job J_1 has been scheduled by algorithm A, $i = 1, 2$. It is clear that the current optimal makespan is $\frac{M_1}{s+1} = \frac{sl_1+l_2}{s+1}$. We consider two cases with respect to the values of l_1 and l_2.

Case 1 $l_1 \geq l_2$. The makespan produced by A is l_1, so

$$\rho \geq \frac{l_1}{\frac{sl_1+l_2}{s+1}} \doteq \rho_1. \tag{2}$$

Then the second and last job $J_2 = \{M_2, R_2\}$ arrives, where $M_2 = l_1 - l_2$ and R_2 has only one task $r_2^1 = (sl_1 + l_2)s$. It is easy to see that the makespan produced by algorithm A is at least $l_1 + sl_1 + l_2 = (s+1)l_1 + l_2$. On the other hand, we find that the optimal makespan is $\frac{l_1-l_2}{s+1} + sl_1 + l_2$ by scheduling all the map tasks in M_2 on the two machines evenly, r_2^1 on σ_1, and all the map tasks in M_1 on σ_2. Thus, we have

$$\rho \geq \frac{(s+1)l_1 + l_2}{\frac{l_1 - l_2}{s+1} + sl_1 + l_2} \doteq \rho_2. \tag{3}$$

If $l_2 = 0$, we have $\rho \geq \frac{s+1}{s} > \frac{\sqrt{s^2+2s+5}+1-s}{2}$ by (2). So we assume $l_2 > 0$ and let $q = \frac{l_1}{l_2}$. From (2) and (3), we have

$$\rho_1 = \frac{q}{\frac{1+sq}{s+1}} = \frac{(s+1)q}{sq+1}$$

and

$$\rho_2 = \frac{1+(s+1)q}{\frac{q-1}{s+1} + 1 + sq} = \frac{(s+1)q+1}{(s^2+s+1)q+s}.$$

It is easy to see that for $q \geq 1$, ρ_1 is increasing and ρ_2 is decreasing. Since $\rho \geq \max\{\rho_1, \rho_2\}$, we establish the equation $\rho_1 = \rho_2$. Solving the equation, we obtain $q = \frac{s+1+\sqrt{s^2+2s+5}}{2}$, so $\rho \geq \frac{\sqrt{s^2+2s+5}+1-s}{2}$.

Case 2 $l_1 \leq l_2$. The makespan produced by A is l_2, so

$$\rho \geq \frac{l_2}{\frac{sl_1+l_2}{s+1}}. \tag{4}$$

Then the second and last job $J_2 = \{M_2, R_2\}$ arrives, where $M_2 = (l_2 - l_1)s$ and R_2 has only one task $r_2^1 = (sl_1 + l_2)s$. It is easy to see that the makespan produced by algorithm A is at least $l_2 + sl_1 + l_2 = sl_1 + 2l_2$. On the other hand, we find that the optimal makespan is $\frac{(l_2-l_1)s}{s+1} + sl_1 + l_2$. Thus,

$$\rho \geq \frac{sl_1 + 2l_2}{\frac{(l_2-l_1)s}{s+1} + sl_1 + l_2}. \tag{5}$$

Using similar arguments in Case 1, we obtain the desired result from (4) and (5). □

For the non-preemptive variant, note that $Q2|online|C_{\max}$ is a special case of our problem $Q2|M(frac)R, online|C_{\max}$, so a lower bound for $Q2|online|C_{\max}$ [4] is also a lower bound for of $Q2|M(frac)R, online|C_{\max}$.

Theorem 2.3 *For the variant $Q2|M(frac)R, online|C_{\max}$, the competitive ratio of any on-line algorithm is at least*

$$\begin{cases} \frac{2s+1}{s+1}, & \text{if } s < \frac{1+\sqrt{5}}{2}; \\ \frac{s+1}{s}, & \text{if } s \geq \frac{1+\sqrt{5}}{2}. \end{cases}$$

3 An optimal on-line algorithm for the preemptive variant

In this section we provide an online algorithm A_1 with a competitive ratio $\alpha = \frac{\sqrt{s^2+2s+5}+1-s}{2} > 1$ for the preemptive variant $Q2|M(frac)R(pmtn), online|C_{\max}$. Let l^i_j denote the completion time of machine σ_i at the moment right after the jth job J_j has been scheduled, $i = 1, 2$.

Before presenting our algorithm A_1, we introduce a useful lemma as follows:

Lemma 3.1 *If* $l^1_j = \frac{\alpha}{1+s-\alpha s} l^2_j$ *for some* $1 \leq j \leq n$*, we have* $\frac{C^{A_1}_j}{C^*_j} \leq \alpha$.

Proof Noting that $\frac{\alpha}{1+s-\alpha s} > 1$, we have $C^{A_1}_j = l^1_j = \frac{\alpha}{1+s-\alpha s} l^2_j$. By (1), we have $C^*_j \geq \frac{P_j}{1+s} = \frac{sl^1_j+l^2_j}{1+s} = \frac{1}{1+s-\alpha s} l^2_j$, so $\frac{C^{A_1}_j}{C^*_j} \leq \alpha$. □

The main idea of our algorithm A_1 is inspired by Lemma 3.1. To achieve the desired competitive ratio, algorithm A_1 always tries to schedule the jobs so that the completion times of the two machines keep the ratio α: $1 + s - \alpha s$. In other words, it only needs to schedule the current job J_j so that $l^1_j = \frac{\alpha}{1+s} P_j$ and $l^2_j = \frac{1+s-\alpha s}{1+s} P_j$. If the ratio cannot be maintained, we conclude that the current job J_j has a very large reduce task r^1_j (the details are provided in the algorithm). In that case, the optimal makespan is also determined by this job, i.e., $C^*_j \geq \frac{p(M_j)}{1+s} + \frac{r^1_j}{s}$, and we show that $\frac{C^{A_1}_j}{C^*_j} \leq \alpha$ still holds.

In our algorithm, we introduce the procedure $P(M_j, R_j)$ to schedule job J_j when the current completion times of the two machines are the same, i.e., $l^1_{j-1} = l^2_{j-1}$.

Procedure $P(M_j, R_j)$

0. Denote $\Delta_1 = \frac{\alpha}{1+s} P_j - l^1_{j-1} - \frac{M_j}{1+s}$ and $\Delta_2 = \frac{1+s-\alpha s}{1+s} P_j - l^1_{j-1} - \frac{M_j}{1+s}$.
1. If $s\Delta_1 > R_j$, schedule the portion $(1 + s - \alpha s)(M_j + R_j - \frac{s(s+1)(\alpha-1)}{1+s-\alpha s} l^1_{j-1})$ (denoted by M^1_j) of the map tasks in M_j on the two machines evenly, and the leftover of the map tasks (denoted by M^2_j) and all the reduce tasks on σ_1 (see Fig. 2a).
2. If $r^1_j < s\Delta_1 \leq R_j$, schedule all the map tasks on the two machines evenly. For the reduce tasks:
   - 2.1 If $r^1_j \leq \Delta_2$, find $k = \min\{q | \sum_{i=1}^{q} r^i_j \geq \Delta_2, 1 \leq q \leq v_j\}$ and partition r^k_j into two parts $r^{k_1}_j$ and $r^{k_2}_j$ such that $\sum_{i=1}^{k-1} r^i_j + r^{k_1}_j = \Delta_2$. Then schedule $r^1_j, \ldots, r^{k-1}_j, r^{k_1}_j$ on σ_2 and $r^{k_2}_j, r^{k+1}_j, \ldots, r^{v_j}_j$ on σ_1. (see Fig. 2b, where $R^1_j = \{r^2_j, \ldots, r^{k-1}_j\}$ and $R^1_j = \{r^{k+1}_j, \ldots, r^{v_j}_j\}$.)
   - 2.2 If $r^1_j > \Delta_2$, partition r^1_j into two parts $r^{1_1}_j$ and $r^{1_2}_j$ such that $r^{1_1}_j = \Delta_2$. Then schedule $r^{1_1}_j$ on σ_2 in the time interval $[l^1_{j-1} + \frac{M_j}{1+s}, \frac{1+s-\alpha s}{1+s} P_j]$ and $r^{1_2}_j$ on σ_1 in the time interval $[\frac{1+s-\alpha s}{1+s} P_j, \frac{1+s-\alpha s}{1+s} P_j + \frac{r^{1_2}_j}{s}]$, and the leftover on σ_1 as early as possible (see Fig. 2c, Where $R^1_j \bigcup R^2_j = R_j \backslash \{r^1_j\}$ and $R^1_j = r^{1_1}_j$).

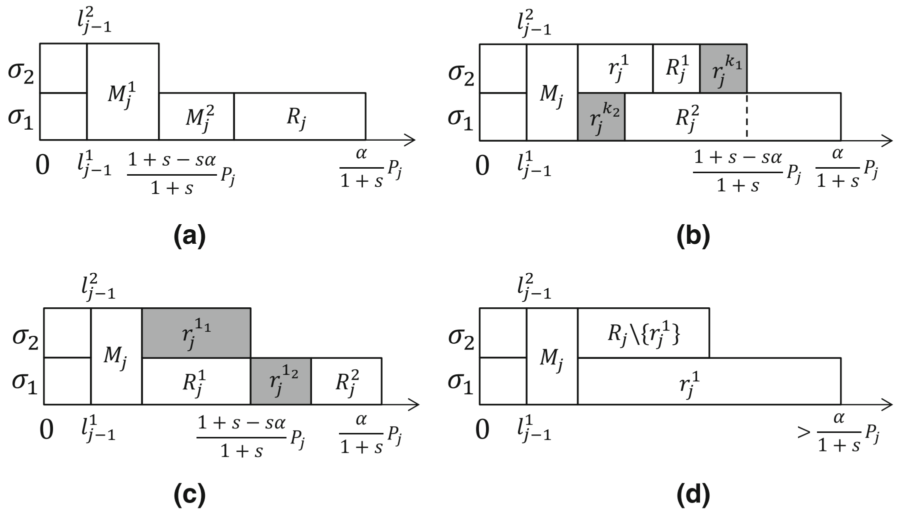


Fig. 2 Assignment of job J_j by procedure P

3. If $s\Delta_1 \le r_j^1$, schedule all the map tasks on the two machines evenly, the largest reduce task r_j^1 on σ_1, and the leftover of the reduce tasks on σ_2 (see Fig. 2d).

Algorithm A_1

1. Let $l_0^1 = l_0^2 = 0$ and invoke procedure $P(M_1, R_1)$ for the first job J_1.
2. For every job J_j, $2 \le j \le n$, denote $\Gamma_1 = \frac{1+s-\alpha s}{1+s}P_j - l_{j-1}^2$, $\Gamma_2 = \frac{\alpha}{1+s}P_j - l_{j-1}^2$, $\Gamma_3 = \frac{1+s-\alpha s}{1+s}P_j - l_{j-1}^1$, and $\Gamma_4 = \frac{\alpha}{1+s}P_j - l_{j-1}^1$. Clearly, $\Gamma_1 < \Gamma_2$ and $\Gamma_3 < \Gamma_4$. If $M_j \le l_{j-1}^1 - l_{j-1}^2$, go to Step 3; otherwise, go to Step 4.
3. For $M_j \le l_{j-1}^1 - l_{j-1}^2$.
   - 3.1 If $M_j + R_j \le \Gamma_1$, schedule all the map tasks and reduce tasks on σ_2.
   - 3.2 If $M_j + r_j^1 \le \Gamma_1 < M_j + R_j$, schedule all the map tasks on σ_2. For the reduce tasks:
     - (a) If $r_j^1 \le \Gamma_3$, schedule r_j^1 on σ_2 in the time interval $[l_{j-1}^1, l_{j-1}^1 + r_j^1]$, and the leftover on σ_2 in the time intervals $[l_{j-1}^2 + M_j, l_{j-1}^1]$ and $[l_{j-1}^1 + r_j^1, \frac{1+s-\alpha s}{1+s}P_j]$, and on σ_1 in the time interval $[l_{j-1}^1, \frac{\alpha}{1+s}P_j]$ (see Fig. 3).
     - (b) If $r_j^1 > \Gamma_3$, schedule r_j^1 on σ_2 in the time interval $[\frac{1+s-\alpha s}{1+s}P_j - r_j^1, \frac{1+s-\alpha s}{1+s}P_j]$, and the leftover on σ_2 in the time intervals $[l_{j-1}^2 + M_j, \frac{1+s-\alpha s}{1+s}P_j - r_j^1]$ and on σ_1 in the time interval $[l_{j-1}^1, \frac{\alpha}{1+s}P_j]$ (see Fig. 4).
   - 3.3 If $M_j + r_j^1 > \Gamma_1$, schedule all the map tasks on σ_2. For the reduce tasks:
     - (a) If $\frac{r_j^1}{s} \le \Gamma_3$, partition r_j^1 into two parts $r_j^{1_1}$ and $r_j^{1_2}$ such that $r_j^{1_1} + \frac{r_j^{1_2}}{s} = \Gamma_3$. Then schedule $r_j^{1_1}$ on σ_2 in the time interval $[l_{j-1}^1, l_{j-1}^1 + r_j^{1_1}]$ and $r_j^{1_2}$ on σ_1 in the time interval$[l_{j-1}^1 + r_j^{1_1}, \frac{1+s-\alpha s}{1+s}P_j]$, and the leftover on σ_2 in

Fig. 3 Step 3.2(a)

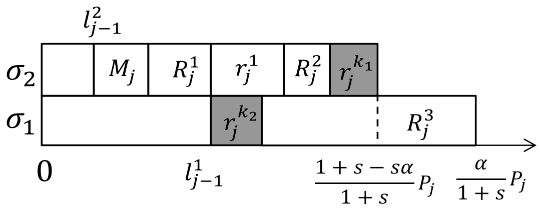
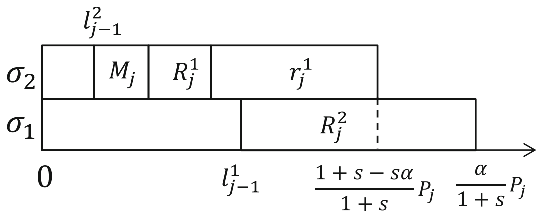


Fig. 4 Step 3.2(b)



the time intervals $[l^2_{j-1}+M_j, l^1_{j-1}]$ and $[l^1_{j-1}+r_j^{1_1}, \frac{1+s-\alpha s}{1+s}P_j]$, and on σ_1 in the time intervals $[l^1_{j-1}, l^1_{j-1}+r_j^{1_1}]$ and $[\frac{1+s-\alpha s}{1+s}P_j, \frac{\alpha}{1+s}P_j]$ (see Fig. 6a).

(b) If $\Gamma_3 < \frac{r^1_j}{s} \leq \Gamma_4$, schedule r^1_j on σ_1 in the time interval $[l^1_{j-1}, l^1_{j-1}+\frac{r^1_j}{s}]$, and the leftover on σ_2 in the time interval $[l^2_{j-1}+M_j, \frac{1+s-\alpha s}{1+s}P_j]$ and on σ_1 in the time interval $[l^1_{j-1}+\frac{r^1_j}{s}, \frac{\alpha}{1+s}P_j]$ (see Fig. 6b).

(c) If $\frac{r^1_j}{s} > \Gamma_4$, partition r^1_j into two parts $r_j^{1_1}$ and $r_j^{1_2}$ such that $l^1_{j-1}+\frac{r_j^{1_2}}{s} = \frac{\alpha}{1+s}P_j$.

(1) If $l^2_{j-1}+M_j+r_j^{1_1} \leq l^1_{j-1}$, schedule $r_j^{1_1}$ on σ_2 in the time interval $[l^1_{j-1}-r_j^{1_1}, l^1_{j-1}]$ and $r_j^{1_2}$ on σ_1 in the time interval $[l^1_{j-1}, \frac{\alpha}{1+s}P_j]$, and the leftover on σ_2 in the time interval $[l^2_{j-1}+M_j, l^1_{j-1}-r_j^{1_1}]$ and $[l^1_{j-1}, \frac{1+s-\alpha s}{1+s}P_j]$ [see Fig. 6(c1)].

(2) If $l^2_{j-1}+M_j+r_j^{1_1} > l^1_{j-1}$, partition r^1_j into two parts $r_j^{1'_1}$ and $r_j^{1'_2}$ such that $l^2_{j-1}+M_j+r_j^{1'_1} = l^1_{j-1}$. Then schedule $r_j^{1'_1}$ on σ_2 in the time interval $[l^1_{j-1}-r_j^{1_1}, l^1_{j-1}]$, $r_j^{1'_2}$ on σ_1 in the time interval $[l^1_{j-1}, l^1_{j-1}+\frac{r_j^{1'_2}}{s}]$, and the leftover on σ_2 at time l^1_{j-1} [see Fig. 6(c2)].

4. For $M_j > l^1_{j-1} - l^2_{j-1}$

4.1 If $M_j + R_j \leq \Gamma_2$, schedule the portion $M^1_j = \Gamma_1$ of M_j on σ_2, and the leftover of the map tasks M^2_j and all the reduce tasks on σ_1 (see Fig. 5).

4.2 If $M_j + R_j > \Gamma_2$, schedule the portion $M^1_j = l^1_{j-1} - l^2_{j-1}$ of the map tasks such that $l^2_{j-2} + M^1_j = l^1_{j-1}$ and denote the remainder of the map tasks as $M'_j = M_j \backslash M^1_j$. Invoke procedure $P(M'_j, R_j)$ for the job $J'_j = \{M'_j, R_j\}$.

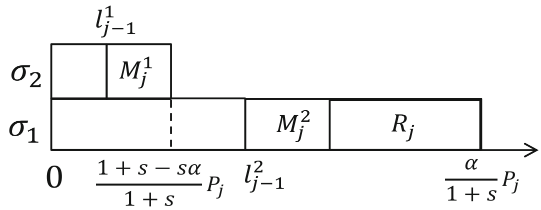


Fig. 5 Step 4.1

Remark By the rules of procedure P and algorithm A_1, we always have $l^1_j \geq \frac{\alpha}{1+s-\alpha s} l^2_j$ after scheduling any job J_j. Specifically,

(i) $l^1_j = \frac{\alpha}{1+s-\alpha s} l^2_j$ after Steps 1 and 2 in procedure P, and Steps 3.2, 3.3(a), 3.3(b), 3.3(c1), and 4.1 in algorithm A_1.
(ii) $l^1_j > \frac{\alpha}{1+s-\alpha s} l^2_j$ after Step 3 in procedure P, and Steps 3.1 and 3.3(c2) in algorithm A_1.

In addition to obtaining the desired competitive ratio, we have to consider the feasibility of the resulting schedule. The assignment of the map tasks is obviously feasible because it is fractional and can be scheduled arbitrarily. To ensure the feasibility of the reduce tasks, the algorithm must follow two rules. One is that the reduce tasks must be processed after the map tasks and the other is that the time slots assigned to the different parts of any preempted reduce task do not overlap.

Theorem 3.2 *For any $1 \leq j \leq n$, we have $l^1_j \geq \frac{\alpha}{1+s-\alpha s} l^2_j$ and $\frac{C^{A_1}_j}{C^*_j} \leq \alpha$.*

Proof We use mathematical induction to prove the result.

First, we consider the case where $j = 1$, i.e., the assignment of the first job J_1. Note that $l^1_{j-1} = l^2_{j-1} = 0$, so the algorithm schedules job J_1 by procedure P.

(i) If J_j is scheduled by Step 1, the assignment of the job is obviously feasible as shown in Fig. 2a. Noting that $l^1_{j-1} = l^2_{j-1} = \frac{P_{j-1}}{1+s}$ and $P_j = M_j + R_j + P_{j-1}$, we have

$$l^2_j = l^2_{j-1} + \frac{M^1_j}{1+s} = l^2_{j-1} + \frac{1+s-\alpha s}{1+s}\left(M_j + R_j - \frac{s(s+1)(\alpha-1)}{1+s-\alpha s} l^2_{j-1}\right) = \frac{1+s-\alpha s}{1+s} P_j$$

and $l^1_j = P_j - \frac{1+s-\alpha s}{1+s} P_j = \frac{\alpha}{1+s} P_j$. Thus, we have $\frac{C^{A_1}_j}{C^*_j} \leq \alpha$ by Lemma 3.1.

(ii) If J_j is scheduled by Step 2. It is easy to verify that $l^1_j = \frac{\alpha}{1+s} P_j$ and $l^2_j = \frac{1+s-\alpha s}{1+s} P_j$ by the rule of the algorithm. If the reduce tasks are scheduled by Step 2.1, we only need to consider the assignment of the preempted task r^k_j. Since $r^{k_2}_j \leq r^k_j \leq r^1_j$ by the assumption that $r^1_j \geq \cdots \geq r^{v_j}_j$, we have $\frac{r^{k_2}_j}{s} \leq r^1_j$. This means that the start time of $r_j^{k_1}$ is not earlier than the finish time of $r_j^{k_2}$ as shown in Fig. 2b. Thus the time slots assigned to $r_j^{k_1}$ and $r_j^{k_2}$ do not overlap. On the other hand, if the reduce tasks are scheduled by Step 2.2, the assignment of the reduce tasks is obviously feasible as shown in Fig. 2c. Thus, the desired result holds.

(iii) If J_j is scheduled by Step 3, feasibility is trivial. From the rule of the algorithm as shown in Fig. 2d and $l_{j-1}^1 = 0$, we have $C_j^{A_1} = l_j^1 = l_{j-1}^1 + \frac{M_j}{1+s} + \frac{r_j^1}{s} = \frac{M_j}{1+s} + \frac{r_j^1}{s} = C_j^*$. By the assumption that $r_j^1 \geq s\Delta_1$, we obtain $r_j^1 \geq \frac{\alpha s}{1+s} P_j - \frac{s}{1+s} M_j$. It follows that $l_j^1 = \frac{M_j}{1+s} + \frac{r_j^1}{s} \geq \frac{\alpha}{1+s} P_j$ and $l_j^2 \leq \frac{1+s-\alpha s}{1+s} P_j$, i.e., $l_j^1 \geq \frac{\alpha}{1+s-\alpha s} l_j^2$.

Hence, the result holds when $j = 1$.

Suppose that the result is true for $j-1$ $(j \geq 2)$, i.e., $l_{j-1}^1 \geq \frac{\alpha}{1+s-\alpha s} l_{j-1}^2$ and $\frac{C_{j-1}^{A_1}}{C_{j-1}^*} \leq \alpha$. We consider the assignment of J_j below.

(I) J_j is scheduled by Step 3.1. Note that all the tasks are scheduled on σ_1, so the schedule is obviously feasible and we have $l_j^2 = l_{j-1}^2 + M_j + R_j \leq l_{j-1}^2 + \Gamma_1 - l_{j-1}^2 \leq \frac{1+s-\alpha s}{1+s} P_j$ by the assumption that $M_j + R_j \leq \Gamma_1 = \frac{1+s-\alpha s}{1+s} P_j - l_{j-1}^2$. So we have $l_j^1 = l_{j-1}^1 \geq \frac{\alpha}{1+s} P_j$, i.e., $l_j^1 \geq \frac{\alpha}{1+s-\alpha s} l_j^2$. It is clear that $C_j^{A_1} = C_{j-1}^{A_1}$ and $C_j^* \geq C_{j-1}^*$. Hence,

$$\frac{C_j^{A_1}}{C_j^*} \leq \frac{C_{j-1}^{A_1}}{C_{j-1}^*} \leq \alpha.$$

(II) J_j is scheduled by Step 3.2. We have $l_j^2 = l_{j-1}^2 + \Gamma_1 = \frac{1+s-\alpha s}{1+s} P_j$ and $l_j^1 = \frac{\alpha}{1+s} P_j$ as shown in Figs. 2 and 3. If the job is scheduled by Step 3.2(a), we only need to consider the assignment of the preempted task r_j^k. Since $r_j^{k_2} \leq r_j^k \leq r_j^1$, we have $\frac{r_j^{k_2}}{s} \leq r_j^1$, which implies that the start time of $r_j^{k_1}$ is not earlier than the finish time of $r_j^{k_2}$, as shown in Fig. 3. If the job is scheduled by Step 3.2(b), the assignment of the job is obviously feasible as shown in Fig. 4.

(III) J_j is scheduled by Steps 3.3(a), 3.3(b), or 3.3(c1). It is not hard to verify that $l_j^1 = \frac{\alpha}{1+s} P_j$ and $l_j^2 = \frac{1+s-\alpha s}{1+s} P_j$ by the rule of the algorithm as shown in Fig. 6. If the job is scheduled by Step 3.3(a), as shown in Fig. 6a, the time slots assigned to R_j^1, R_j^2, R_j^3, and R_j^4 do not overlap. If the job is scheduled by Step 3.3(b), we see that the time slots assigned to R_j^1 and R_j^2 do not overlap because the completion time of r_j^1 is greater than $\frac{1+s-\alpha s}{1+s} P_j$. If the job is scheduled by Step 3.3(c1), the assignment of the job is obviously feasible as shown in Fig. 6(c1).

(IV) J_j is scheduled by Step 3.3(c2). We have $C_j^{A_1} = l_{j-1}^1 + \frac{r_j^{1'_2}}{s} > \frac{\alpha}{1+s} P_j$ as can been seen in Fig. 6(c2) and $C_j^* \geq \frac{M_j}{1+s} + \frac{r_j^1}{s}$ by Lemma 2.1. Since $l_{j-1}^1 = l_{j-1}^2 + M_j + r_j^{1'_1}$,

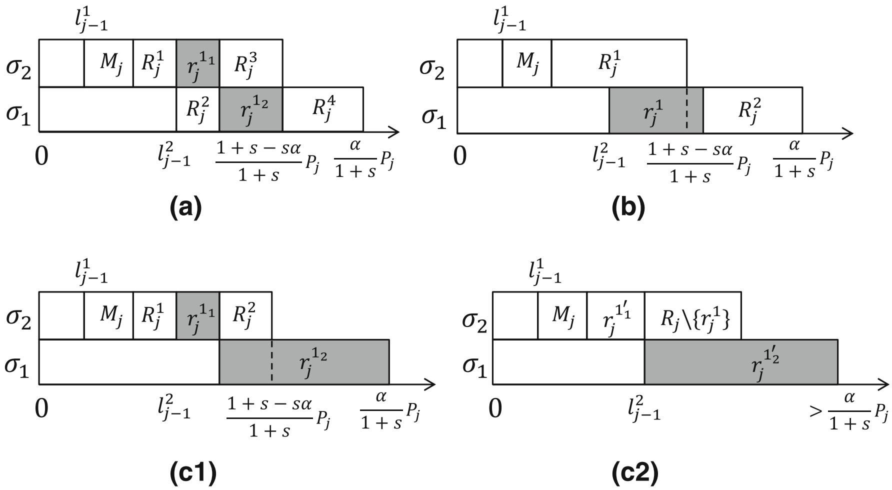


Fig. 6 Step 3.3

$$P_j = M_j + R_j + sl_{j-1}^1 + l_{j-1}^2 \geq M_j + r_j^{1'_1} + r_j^{1'_2} + sl_{j-1}^1 + l_{j-1}^2$$
$$= (s+1)l_{j-1}^1 + r_j^{1'_2}.$$

Thus,

$$\frac{r_j^{1'_2}}{s} > \frac{(\alpha-1)s(s+1)}{1+s-\alpha s} l_{j-1}^1.$$

With the induction assumption that $l_{j-1}^2 \leq \frac{1+s-\alpha s}{\alpha} l_{j-1}^1$, we have

$$\frac{C_j^{A_1}}{C_j^*} \leq \frac{l_{j-1}^1 + \frac{r_j^{1'_2}}{s}}{\frac{M_j}{1+s} + \frac{r_j^1}{s}} \leq \frac{l_{j-1}^1 + \frac{r_j^{1'_2}}{s}}{\frac{l_{j-1}^1 - l_{j-1}^2}{1+s} + \frac{r_j^{1'_2}}{s} + \frac{r_j^{1'_1}}{s} - \frac{r_j^{1'_1}}{1+s}}$$
$$< \frac{l_{j-1}^1 + \frac{r_j^{1'_2}}{s}}{\frac{l_{j-1}^1 - l_{j-1}^2}{1+s} + \frac{r_j^{1'_2}}{s}} \leq \frac{l_{j-1}^1 + \frac{(\alpha-1)(s+1)}{1+s-\alpha s} l_{j-1}^1}{\frac{(1-\frac{1+s-\alpha s}{\alpha})l_{j-1}^1}{1+s} + \frac{(\alpha-1)(s+1)}{1+s-\alpha s} l_{j-1}^1} = \alpha.$$

Finally, we have $l_j^1 \geq \frac{\alpha}{1+s-\alpha s} l_j^2$ from the fact that $l_j^1 > \frac{\alpha}{1+s} P_j$ and $l_j^2 < \frac{1+s-\alpha s}{1+s} P_j$.

(V) J_j is scheduled by Step 4.1, feasibility is trivial. We have $l_j^2 = l_{j-1}^2 + \Gamma_1 = \frac{1+s-\alpha s}{1+s} P_j$ and $l_j^1 = \frac{\alpha}{1+s} P_j$ as shown in Fig. 6. Thus, the result is true.

(VI) J_j is scheduled by Step 4.2. From the algorithm, we see that the loads of the two machines are the same after scheduling the portion $M_j^1 = l_{j-1}^1 - l_{j-1}^2$. Then the leftover $J_j' = \{M_j', R_j\}$ is scheduled by procedure P. If J_j' is scheduled by Step 1 (or Step 2) in procedure P, then we obtain the desired result by the same discussion in (i) [or (ii)]. So we only need to consider the case where J'_j is scheduled by Step 3.

It is easy to obtain that $C_j^{A_1} = l_{j-1}^1 + \frac{M'_j}{1+s} + \frac{r_j^1}{s} > \frac{\alpha}{1+s} P_j$ by the assumption that $r_j^1 \geq s\Delta_1$ and $C_j^* \geq \frac{M_j}{1+s} + \frac{r_j^1}{s}$ by Lemma 2.1. Since $M_j^1 = l_{j-1}^1 - l_{j-1}^2$, we have

$$P_j = M_j^1 + M'_j + R_j + s l_{j-1}^1 + l_{j-1}^2 \geq r_j^1 + (s+1) l_{j-1}^1.$$

So we obtain

$$r_j^1 \geq \frac{(\alpha-1)s(s+1)}{1+s-\alpha s} l_{j-1}^1 > \frac{(\alpha-1)s(s+1)}{1+s-\alpha s} l_{j-1}^1.$$

By the induction assumption that $l_{j-1}^2 \leq \frac{1+s-\alpha s}{\alpha} l_{j-1}^1$, we have

$$\frac{C_j^{A_1}}{C_j^*} \leq \frac{l_{j-1}^1 + \frac{M'_j}{1+s} + \frac{r_j^1}{s}}{\frac{M_j^1}{1+s} + \frac{M'_j}{1+s} + \frac{r_j^1}{s}} \leq \frac{l_{j-1}^1 + \frac{r_j^1}{s}}{\frac{l_{j-1}^1 - l_{j-1}^2}{1+s} + \frac{r_j^1}{s}} \leq \frac{l_{j-1}^1 + \frac{(\alpha-1)(s+1)}{1+s-\alpha s} l_{j-1}^1}{\frac{(1-\frac{1+s-\alpha s}{\alpha}) l_{j-1}^1}{1+s} + \frac{(\alpha-1)(s+1)}{1+s-\alpha s} l_{j-1}^1} = \alpha.$$

Clearly, we have $l_j^1 \geq \frac{\alpha}{1+s-\alpha s} l_j^2$ in this case.

In sum, we conclude that the result holds for any $j \geq 2$ and the proof is complete. □

4 An optimal online algorithm for the non-preemptive variant

In this section we provide an optimal online algorithm A_2 for the variant $Q2|M(frac)R$, $onine|C_{\max}$. The main idea of the algorithm is to schedule the map tasks as early as possible and then schedule all the reduce tasks by the List Scheduling (LS) rule.

Algorithm A_2

1. For every job J_j, $1 \leq j \leq n$, schedule the map tasks as early as possible.
2. Schedule the reduce tasks by the LS rule.
3. Output the schedule.

Theorem 4.1 *For the variant $Q2|M(frac)R, online|C_{\max}$, the competitive ratio of algorithm A_2 is*

$$\gamma = \begin{cases} \frac{2s+1}{s+1}, & if\, s < \frac{1+\sqrt{5}}{2}; \\ \frac{s+1}{s}, & if\, s \geq \frac{1+\sqrt{5}}{2} \end{cases}.$$

Proof The proof is similar to that for the problem $Q2|online|C_{\max}$. Let r be the last finished reduce task and L_j be the completion time of the machine σ_i before scheduling the reduce task r, $i = 1, 2$. By the rule of Algorithm A_2, we have

$$C^{A_2} = \min\left\{L_1 + \frac{r}{s}, L_2 + r\right\}. \tag{6}$$

It is clear that

$$P_n = sL_1 + L_2 + r \leq (1+s)C^* \tag{7}$$

and

$$\frac{r}{s} \leq C^*. \tag{8}$$

From (6), we have $C^{A_2} \leq L_1 + \frac{r}{s}$ and $C^{A_2} \leq L_2 + r$. Combining with (7) and (8), we get

$$(s+1)C^{A_2} - r \leq (s+1)C^*.$$

Thus, we have $\frac{C^{A_2}}{C^*} \leq \frac{2s+1}{s+1}$.

By $C^{A_2} \leq L_1 + \frac{r}{s}$ and (7), we have

$$C^{A_2} \leq \frac{sL_1 + r}{s} \leq \frac{sL_1 + L_2 + r}{s} \leq \frac{(1+s)C^*}{s}$$

so $\frac{C^{A_2}}{C^*} \leq \frac{s+1}{s}$.

Hence,

$$\frac{C^{A_2}}{C^*} \leq \min\left\{\frac{2s+1}{s+1}, \frac{s+1}{s}\right\} = \begin{cases} \frac{2s+1}{s+1}, & \text{if } s < \frac{1+\sqrt{5}}{2}; \\ \frac{s+1}{s}, & \text{if } s \geq \frac{1+\sqrt{5}}{2}. \end{cases} \qquad \square$$

From the proof and the result, we conclude that the problem $Q2|M(frac)R, online|C_{\max}$ is not harder than the classical problem $Q2|online|C_{\max}$.

5 Conclusion

In this paper we study online MapReduce scheduling on two uniform machines to minimize the makespan. Under the assumption that the map tasks are fractional, we consider the cases involving preemptive and non-preemptive reduce tasks. For both variants, we derive lower bounds and optimal online algorithms. We find that the preemptive variant is more complicated than the classical preemptive problem $Q2|pmtn, online|C_{\max}$, while the non-preemptive variant is the same as complicated as the classical problem $Q2|online|C_{\max}$. Possible future research directions include studying the general case with $m \geq 3$ machines and online algorithms with partial information.

Acknowledgements Jiang was supported in part by the National Natural Science Foundation of China (Grant No. 11571013). Cheng was supported in part by The Hong Kong Polytechnic University under the Fung Yiu King—Wing Hang Bank Endowed Professorship in Business Administration. Ji was supported in part by Zhejiang Provincial Natural Science Foundation of China (Grant No. LR15G010001).

References

1. Chang, H., Kodialam, M., Kompella, R.R., Lakshman, T.V.: Scheduling in MapReduce-like systems for fast completion time. In: Proceedings of INFOCOM'14, pp. 3074–3082 (2015)
2. Chen, C., Xu, Y., Zhu, Y., Sun, C.: Online MapReduce scheduling problem of minimizing the makespan. J. Comb. Optim. **33**(2), 590–608 (2017)
3. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. Proc. Oper. Syst. Des. Implement. **51**(1), 107–113 (2004)
4. Epstein, L., Noga, J., Seiden, S., Sgall, J., Woeginger, G.: Randomized on-line scheduling on two uniform machines. J. Sched. **4**, 71–92 (2001)
5. Huang, J., Zheng, F., Xu, Y., Liu, M.: Online MapReduce processing on two identical parallel machines. J. Combin. Optim. **35**(1), 216–223 (2018)
6. Jiang, Y., Zhu, Y., Wu, W., Li, D.: Makespan minimization for MapReduce systems with different servers. Future Gener. Comput. Syst. **67**, 13–21 (2017)
7. Jiang, Y., Zhou, W., Zhou, P.: An optimal preemptive algorithm for online MapReduce scheduling on two parallel machines. Asia-Pac. J. Oper. Res. **35**(3), 1850013 (2018)
8. Luo, T., Zhu, Y., Wu, W., Xu, Y., Du, D.: Online makespan minimization in MapReduce-like systems with complex reduce tasks. Optim. Lett. **11**, 271–277 (2017)
9. Le, Y., Liu, J., Ergun, F., Wang, D.: Online load balancing for MapReduce with skewed data input. In: Proceeding of INFOCOM'14, pp. 2004–2012 (2014)
10. Moseley, B., Dasgupta, A., Kumar, R., Sarlós, T.: On scheduling in Map-Reduce and flowshops. In: Proceedings of the Twenty-Third Annual ACM Symposium on Parallelism in Algorithms and Architectures, pp. 289–298 (2011)
11. Wen, J., Du, D.: Preemptive on-line scheduling for two uniform processors. Oper. Res. Lett. **23**, 113–116 (1998)
12. Zheng, Y., Shroff, N.B., Sinha, P.: A new analytical technique for designing provably efficient MapReduce schedulers. In: Proceeding of INFOCOM'13, pp. 1600–1608 (2013)
13. Zhu, Y., Jiang, Y., Wu, W., Ding, L., Teredesai, A., Li, D., Lee, W.: Minimizing makespan and total completion time in MapReduce-like systems. In: Proceeding of INFOCOM'14, pp. 2166–2174 (2014)