Supplemental Material: Dynamic Job Ordering and Slot Configurations for MapReduce Workloads

Shanjiang Tang, Bu-Sung Lee, Bingsheng He

Abstract—MapReduce is a popular parallel computing paradigm for large-scale data processing in clusters and data centers. A MapReduce workload generally contains a set of jobs, each of which consists of multiple map tasks followed by multiple reduce tasks. Due to 1) that map tasks can only run in map slots and reduce tasks can only run in reduce slots, and 2) the general execution constraints that map tasks are executed before reduce tasks, different job execution orders and map/reduce slot configurations for a MapReduce workload have significantly different performance and system utilization. This paper proposes two classes of algorithms to minimize the *makespan* and the *total completion time* for an offline MapReduce workload. Our first class of algorithms focuses on the job ordering optimization for a MapReduce workload under a given map/reduce slot configuration. In contrast, our second class of algorithms considers the scenario that we can perform optimization for map/reduce slot configuration for a MapReduce workload. We perform simulations as well as experiments on Amazon EC2 and show that our proposed algorithms produce results that are up to $15\% \sim 80\%$ better than currently unoptimized Hadoop, leading to significant reductions in running time in practice.

Index Terms—MapReduce, Hadoop, Flow-shops, Scheduling algorithm, Job ordering.

Appendix A
Proof of Lemma 1

Lemma 1. *For C_{max} in Formula (3) of* Main File, *(1). if $C_{max} = \sum_{i=1}^{k_0} T_i^{\mathcal{M}} + \sum_{i=k_0}^{n} T_i^{\mathcal{R}}$, $(1 \leqslant k_0 \leqslant n)$, there must be $X_{k'} = 0$ for all $k_0 < k' \leqslant n$; (2). Reversely, if $X_{k_0} > 0$ and $X_{k'} = 0$ for all $k_0 < k' \leqslant n$, there must be $C_{max} = \sum_{i=1}^{k_0} T_i^{\mathcal{M}} + \sum_{i=k_0}^{n} T_i^{\mathcal{R}}$, $(1 \leqslant k_0 \leqslant n)$.*

Proof: Let $f(k,n) = \sum_{i=1}^{k} T_i^{\mathcal{M}} + \sum_{i=k}^{n} T_i^{\mathcal{R}}$. Our proof consists of the following two parts:

(i). $C_{max} = f(k_0, n) \Rightarrow X_{k'} = 0$ for all $k_0 < k' \leqslant n$.
Proof: Let's do the proof by contradiction. Suppose there is a $k_1 (k_0 < k_1 \leqslant n)$ such that $X_{k_1} > 0$. According to Formula (1) and (3) of *Main File*, it holds $\sum_{i=1}^{k_1} T_i^{\mathcal{M}} > \sum_{i=1}^{k_1-1} (T_i^{\mathcal{R}} + X_i) = \max_{1 \leqslant k \leqslant k_1-1}\{f(k, k_1-1)\} \geqslant f(k_0, k_1-1)$. Therefore, we have $f(k_1, n) = \sum_{i=1}^{k_1} T_i^{\mathcal{M}} + \sum_{i=k_1}^{n} T_i^{\mathcal{R}} > f(k_0, k_1-1) + \sum_{i=k_1}^{n} T_i^{\mathcal{R}} = f(k_0, n)$, which violates the concept of C_{max}. That is, the assumption is wrong and therefore the proposition is true.

(ii). $X_{k_0} > 0 \& X_{k'} = 0$ for all $k_0 < k' \leqslant n$. $\Rightarrow C_{max} = f(k_0, n)$.
Proof: According to Equation (3) of *Main File*, $C_{max} = \sum_{i=1}^{n} (X_i + T_i^{\mathcal{R}}) = \sum_{i=1}^{k_0} (X_i + T_i^{\mathcal{R}}) + \sum_{i=k_0+1}^{n} T_i^{\mathcal{R}}$. Thus our work is equivalent to prove $\sum_{i=1}^{k_0} (X_i + T_i^{\mathcal{R}}) = \max_{1 \leqslant k \leqslant k_0}\{f(k, k_0)\} = f(k_0, k_0)$. That is to prove $\max_{1 \leqslant k \leqslant k_0-1}\{f(k, k_0)\} < f(k_0, k_0)$. Noting that $X_{k_0} > 0$, based on Equation (1) of *Main File*, we have $\sum_{i=1}^{k_0} T_i^{\mathcal{M}} > \sum_{i=1}^{k_0-1} (T_i^{\mathcal{R}} + X_i)$. Thus $f(k_0, k_0) = \sum_{i=1}^{k_0} T_i^{\mathcal{M}} + \sum_{i=k_0}^{k_0} T_i^{\mathcal{R}} > \sum_{i=1}^{k_0-1} (T_i^{\mathcal{R}} + X_i) + \sum_{i=k_0}^{k_0} T_i^{\mathcal{R}} = \max_{1 \leqslant k \leqslant k_0-1}\{f(k, k_0-1)\} + T_i^{\mathcal{R}} = \max_{1 \leqslant k \leqslant k_0-1}\{f(k, k_0)\}$. Therefore, the proposition is true.

In summary, our proof completes by combining **(i)** and **(ii)**. □

• *S.J. Tang, B.S. Lee, B.S. He are with the School of Computer Engineering, Nanyang Technological University, Singapore.*
E-mail: {stang5, ebslee, bshe}@ntu.edu.sg.

Appendix B
Proof of Lemma 2

Lemma 2. *Given a job order ϕ, there is a upper bound* makespan *denoted by $\widehat{C}_{max}$ (i.e. $C_{max} \leqslant \widehat{C}_{max}$) for the generalized case as follows:*

$$\widehat{C}_{max} = \max_{1 \leqslant k \leqslant n} \Big\{ \sum_{i=1}^{k} T_i^{\mathcal{M}} + \max_{1 \leqslant i \leqslant k} \{\widehat{t}_i^{\mathcal{M}}\} + \sum_{i=k}^{n} T_i^{\mathcal{R}} + \max_{1 \leqslant i \leqslant n} \{\widehat{t}_i^{\mathcal{R}}\} \Big\}.$$

where $T_i^{\mathcal{M}} = \frac{\sum_{j=1}^{|J_i^{\mathcal{M}}|} t_{i,j}^{\mathcal{M}}}{|\mathcal{S}^{\mathcal{M}}|}$, $T_i^{\mathcal{R}} = \frac{\sum_{j=1}^{|J_i^{\mathcal{R}}|} t_{i,j}^{\mathcal{M}}}{|\mathcal{S}^{\mathcal{R}}|}$, $\widehat{t}_i^{\mathcal{M}} = \max_{1 \leqslant j \leqslant |J_i^{\mathcal{M}}|} \widehat{t}_{i,j}^{\mathcal{M}}$ *and* $\widehat{t}_i^{\mathcal{R}} = \max_{1 \leqslant j \leqslant |J_i^{\mathcal{R}}|} \widehat{t}_{i,j}^{\mathcal{R}}$.

Proof: Lemma 1 gives us an important implication regarding the *makespan* estimation for the simplified case. That is, finding a job J_{k_0} such that $X_{k_0} > 0$ and $X_{k'} = 0$ (i.e., no idle period in the reduce phase after the reduce tasks of J_{k_0} starts) for all $k_0 < k' \leqslant n$. Then the *makespan* is equal to $\sum_{i=1}^{k_0} T_i^{\mathcal{M}} + \sum_{i=k_0}^{n} T_i^{\mathcal{R}}$. It is also applicable for the generalized case.

Let $c_i^{\mathcal{M}}$ be the completion time at the map phase and $c_i^{\mathcal{R}}$ be the completion time at the reduce phase for the generalized case. Then we have $c_i^{\mathcal{M}} < \sum_{j=1}^{i} \frac{\sum_{j=1}^{|J_i^{\mathcal{M}}|} t_{i,j}^{\mathcal{M}}}{|\mathcal{S}^{\mathcal{M}}|} + \max_{1 \leqslant j \leqslant i}\{\widehat{t}_j^{\mathcal{M}}\}$. For the generalized case, there are two types of idle periods in the reduce phase, i.e., *partial idle period (PIP)* (e.g. X_3 in Figure 4 for Job J_3) and *full idle period (FIP)* (e.g., X_4 in

Figure 4 for Job J_4), unlike the simplified case that has *FIP* only. Moreover, it is worth noting that for the simplified case, the job execution order is the same between in the map phase and in reduce phase. Whereas it does not always hold for the generalized case. In Figure 4, for example, the reduce tasks of J_2 start earlier than that of J_1 although the map tasks of J_1 start earlier in the map phase. Therefore, we need to consider the following two cases:

(1). Suppose the execution order in the map phase is the same as in the reduce phase for the generalized case. Let's assume that Job J_{k_0} is the splitting point that $X_{k_0} > 0$ (either *FIP* or *FIP*) and $X_{k'} = 0$ (no idle period) for all the remaining jobs $J_{k'}(k_0 < k' \leqslant n)$. Then we have $C_{max} < c_{k_0}^{\mathcal{M}} + \sum_{i=k_0}^{n} \frac{\sum_{j=1}^{|J_i^{\mathcal{M}}|} t_{i,j}^{\mathcal{M}}}{|\mathcal{S}^{\mathcal{R}}|} + \max_{1 \leqslant i \leqslant n}\{\widehat{t}_i^{\mathcal{R}}\} < \sum_{i=1}^{k_0} \frac{\sum_{j=1}^{|J_i^{\mathcal{M}}|} t_{i,j}^{\mathcal{M}}}{|\mathcal{S}^{\mathcal{M}}|} + \max_{1 \leqslant i \leqslant k_0}\{\widehat{t}_i^{\mathcal{M}}\} + \sum_{i=k_0}^{n} \frac{\sum_{j=1}^{|J_i^{\mathcal{R}}|} t_{i,j}^{\mathcal{R}}}{|\mathcal{S}^{\mathcal{R}}|} + \max_{1 \leqslant i \leqslant n}\{\widehat{t}_i^{\mathcal{R}}\} \leqslant \max_{1 \leqslant k \leqslant n}\Big\{\sum_{i=1}^{k} \frac{\sum_{j=1}^{|J_i^{\mathcal{M}}|} t_{i,j}^{\mathcal{M}}}{|\mathcal{S}^{\mathcal{M}}|} + \max_{1 \leqslant i \leqslant k}\{\widehat{t}_i^{\mathcal{M}}\} + \sum_{i=k}^{n} \frac{\sum_{j=1}^{|J_i^{\mathcal{R}}|} t_{i,j}^{\mathcal{R}}}{|\mathcal{S}^{\mathcal{R}}|} + \max_{1 \leqslant i \leqslant n}\{\widehat{t}_i^{\mathcal{R}}\}\Big\}$.

(2). On the other hand, it is possible that there are out of order jobs in the reduce phase for the generalized case (e.g. job order in Figure 4). Note that the cause of the out of order at the reduce phase is due to the map tasks of later submitted job finish first. It can have the reduce tasks begin earlier in the reduce phase, making the maximum completion time for a batch of jobs less than that of the ordered case. In other words, if there is an out of order interval from the i^{th} job to the j^{th} job at the reduce phase, we can have $C_{max}^{j} \leqslant C_{max}^{'j}$, where C_{max}^{j} denotes the maximum completion time for j jobs in the out of order case and $C_{max}^{'j}$ denotes the maximum completion time for j jobs in the ordered case. It may in turn cause later jobs to complete earlier (i.e. $C_{max}^{k} \leqslant C_{max}^{'k}$ for $k > j$). Hence, we have $C_{max} < \max_{1 \leqslant k \leqslant n}\Big\{\sum_{i=1}^{k} \frac{\sum_{j=1}^{|J_i^{\mathcal{M}}|} t_{i,j}^{\mathcal{M}}}{|\mathcal{S}^{\mathcal{M}}|} + \max_{1 \leqslant i \leqslant k}\{\widehat{t}_i^{\mathcal{M}}\} + \sum_{i=k}^{n} \frac{\sum_{j=1}^{|J_i^{\mathcal{R}}|} t_{i,j}^{\mathcal{R}}}{|\mathcal{S}^{\mathcal{R}}|} + \max_{1 \leqslant i \leqslant n}\{\widehat{t}_i^{\mathcal{R}}\}\Big\}$ for this case, based on the case (1).

Finally, our proof completes based on (1) and (2).

□

APPENDIX C
PROOF OF THEOREM 2

Theorem 2. *Suppose ϕ is the optimal job order whose makespan is C_{max}^{opt}, based on* Johnson's Rule *for the two-stage flow shop with one processor per stage. The worst-case job order ϕ*, whose makespan is C_{max}^{wst}, can be obtained by simply reversing ϕ.*

Proof: Consider Formula (3) of *Main File*, it can be rewritten as follows:

$$C_{max} = \max_{1 \leqslant k \leqslant n} \{T_k\}, \tag{1}$$

where

$$T_k = \sum_{i=1}^{k} T_i^{\mathcal{M}} + \sum_{i=k}^{n} T_i^{\mathcal{R}}.$$

Given a job order ϕ, let's define

$$F(\phi) = C_{max} = \max_{1 \leqslant k \leqslant n} \{T_k\}. \tag{2}$$

Therefore, our proof is equivalent to finding a job order ϕ* such that $F(\phi^*) \geqslant F(\phi)$ for any ϕ.

Let's first consider a job order ϕ' constructed by interchanging J_j and J_{j+1} in ϕ. Then we have

$$F(\phi') = \max_{1 \leqslant k \leqslant n} \{T_k'\}. \tag{3}$$

where

$$T_k' = \sum_{i=1}^{k} T_i^{'\mathcal{M}} + \sum_{i=k}^{n} T_i^{'\mathcal{R}}$$

$$T_i^{'\mathcal{M}} = \begin{cases} T_i^{\mathcal{M}} & (i \neq j \wedge i \neq j+1) \\ T_{j+1}^{\mathcal{M}} & (i = j) \\ T_j^{\mathcal{M}} & (i = j+1) \end{cases}$$

$$T_i^{'\mathcal{R}} = \begin{cases} T_i^{\mathcal{R}} & (i \neq j \wedge i \neq j+1) \\ T_{j+1}^{\mathcal{R}} & (i = j) \\ T_j^{\mathcal{R}} & (i = j+1). \end{cases}$$

Then it holds

$$T_k' = T_k, \qquad (k \neq j \wedge k \neq j+1). \tag{4}$$

Note that $F(\phi') \leqslant F(\phi)$ if $\max\{T_j, T_{j+1}\} \geqslant \max\{T_j', T_{j+1}'\}$, otherwise $F(\phi') > F(\phi)$. To make $F(\phi^*) \geqslant F(\phi)$ be true for any ϕ, we only need to make sure

$$\max\{T_j, T_{j+1}\} \geqslant \max\{T_j', T_{j+1}'\} \tag{5}$$

for all $1 \leqslant j < n$ in ϕ*. Moreover, by subtracting $\sum_{i=1}^{j+1} T_i^{\mathcal{M}} + \sum_{i=j}^{n} T_i^{\mathcal{R}}$ from each term in Formula (5), it turns out to be

$$\max\{-T_{j+1}^{\mathcal{M}}, -T_j^{\mathcal{R}}\} \geqslant \max\{-T_j^{\mathcal{M}}, -T_{j+1}^{\mathcal{R}}\} \tag{6}$$

or equivalent to

$$\min\{T_j^{\mathcal{M}}, T_{j+1}^{\mathcal{R}}\} \geqslant \min\{T_{j+1}^{\mathcal{M}}, T_j^{\mathcal{R}}\}. \tag{7}$$

To satisfy Formula (7) for any two adjacent jobs J_j and J_{j+1} in ϕ*, let's consider the conditions for the following possible cases:

①. When $T_j^{\mathcal{M}} \geqslant T_j^{\mathcal{R}}$ for Job J_j and $T_{j+1}^{\mathcal{M}} \leqslant T_{j+1}^{\mathcal{R}}$ for Job J_{j+1}, Formula (7) holds in this case without any further condition.

②. When $T_j^{\mathcal{M}} \leqslant T_j^{\mathcal{R}}$ for Job J_j and $T_{j+1}^{\mathcal{M}} \geqslant T_{j+1}^{\mathcal{R}}$ for Job J_{j+1}, we have $\min\{T_j^{\mathcal{M}}, T_{j+1}^{\mathcal{R}}\} \leqslant \min\{T_{j+1}^{\mathcal{M}}, T_j^{\mathcal{R}}\}$, which violates Formula (7). Hence this case is impossible in ϕ*.

③. When $T_j^{\mathcal{M}} \geqslant T_j^{\mathcal{R}}$ for Job J_j and $T_{j+1}^{\mathcal{M}} \geqslant T_{j+1}^{\mathcal{R}}$ for Job J_{j+1}, to satisfy Formula (7), we should keep the following hold$T_j^{\mathcal{R}} \leqslant T_{j+1}^{\mathcal{R}}$.

④. When $T_j^{\mathcal{M}} \leqslant T_j^{\mathcal{R}}$ for Job J_j and $T_{j+1}^{\mathcal{M}} \leqslant T_{j+1}^{\mathcal{R}}$ for Job J_{j+1}, to satisfy Formula (7), we should guarantee $T_j^{\mathcal{M}} \geqslant T_{j+1}^{\mathcal{M}}$.

These give us guidance to find ϕ* in the following way: Partition the jobs set J into two disjoint sub-sets J_A and J_B. Set J_A contains those jobs J_i whose $T_i^{\mathcal{M}} > T_i^{\mathcal{R}}$. Set J_B contains the remaining jobs. Sort jobs in J_A in non-decreasing order of $T_i^{\mathcal{R}}$, according to ③. Sort jobs in J_B in non-increasing order of $T_i^{\mathcal{M}}$, according to ④. Finally, ϕ* is formed by appending the sorted J_B to the end of J_A according to ①②.

Note that the resultant ϕ^* of the above job ordering rule is just equivalent to reversing the order of the result based on *Johnson's Rule*. Hence Theorem 2 is true. □

Appendix D
Proof of Lemma 3

Lemma 3. *Let $\breve{\phi}^*$ denote the job order produced by* MK_JR. *Let $\widehat{\phi}^*$ be the reversing order of $\breve{\phi}^*$. Then there is a lower bound makespan denoted by $\breve{C}^*_{max}$, as well as a upper bound makespan denoted by $\widehat{C}^*_{max}$, for all job orders Φ. Particularly, $\breve{C}^*_{max}$ is estimated with regard to $\breve{\phi}^*$ by using Formula (3) of* Main File. *$\widehat{C}^*_{max}$ is estimated with regard to $\widehat{\phi}^*$ with the Formula:*

$$\widehat{C}^*_{max} = \max_{1\leqslant k\leqslant n}\Big\{\sum_{i=1}^{k} T_i^{\mathcal{M}} + \max_{1\leqslant i\leqslant n}\{\widehat{t}_i^{\mathcal{M}}\} + \sum_{i=k}^{n} T_i^{\mathcal{R}} + \max_{1\leqslant i\leqslant n}\{\widehat{t}_i^{\mathcal{R}}\}\Big\}.$$

Proof: Let ϕ denote an arbitrary job order. Our proof makes up of the following two parts:

(i). $C_{max} \geqslant \breve{C}^*_{max}$.

Proof: Let's consider the following two scenarios:

(1). There is a scenario that the job execution order is consistent between the map phase and reduce phase for the generalized case, i.e, $c_j^{\mathcal{M}} \leqslant c_k^{\mathcal{M}}$ if J_j is submitted before J_k. We can construct a simplified case regarding ϕ with $(T_i^{\mathcal{M}}, T_i^{\mathcal{R}}) = (\frac{\sum_{j=1}^{|J_i^{\mathcal{M}}|} t_{i,j}^{\mathcal{M}}}{|\mathcal{S}^{\mathcal{M}}|}, \frac{\sum_{j=1}^{|J_i^{\mathcal{R}}|} t_{i,j}^{\mathcal{R}}}{|\mathcal{S}^{\mathcal{R}}|})$ for each job J_i. Let $\breve{C}_{max}$ denote the makespan for the simplified case. Let $\breve{c}_i^{\mathcal{M}}$ and $\breve{c}_i^{\mathcal{R}}$ denote the completion time at the map phase and reduce phase for the simplified case respectively. Then we have $\breve{c}_i^{\mathcal{M}} = \frac{\sum_{1\leqslant j\leqslant i}\sum_{k=1}^{|J_j^{\mathcal{M}}|} t_{j,k}^{\mathcal{M}}}{|\mathcal{S}^{\mathcal{M}}|}$. However, for the generalized case (See Figure 4), due to the *non-divisible* condition for some jobs, we have $c_i^{\mathcal{M}} \geqslant \frac{\sum_{1\leqslant j\leqslant i}\sum_{k=1}^{|J_j^{\mathcal{M}}|} t_{j,k}^{\mathcal{M}}}{|\mathcal{S}^{\mathcal{M}}|}$ by considering the possible tail (e.g. a map task tail from 24 to 28 time unit for job J_4 in Figure 4). It reveals that for job J_i of the simplified case, its map tasks complete no later than those of the corresponding job of the generalized case at the map phase. Moreover, even though we let $\breve{c}_i^{\mathcal{M}} = c_i^{\mathcal{M}}$ for each job J_i, we still have $\breve{c}_i^{\mathcal{R}} \leqslant c_i^{\mathcal{R}}$ by considering the possible tail (e.g. a reduce task tail from 36 to 44 time unit for job J_4 in Figure 4) at the reduce phase. Note that $c_i = c_i^{\mathcal{R}}$ and $\breve{c}_i = \breve{c}_i^{\mathcal{R}}$, where $\breve{c}_i$ denotes the completion time for job J_i of the simplified case. Therefore, we have $C_{max} \geqslant \breve{C}_{max}$. On the other hand, we can obtain an optimal job order $\breve{\phi}^*$ for makespan $\breve{C}^*_{max}$ in the simplified case with *MK_JR*, in terms of *Johnson's Rule*. Then we have $\breve{C}_{max} \geqslant \breve{C}^*_{max}$. So it holds $C_{max} \geqslant \breve{C}^*_{max}$.

(2). Besides, there is another scenario that the job execution is out of order between map phase and reduce phase for the generalized case (e.g. an out of order execution between J_1 and J_2 in the reduce phase in Figure 4). We can construct a simplified case based on the job execution order (denoted as $\phi^{'}$) of the reduce phase of the generalized case, with $(T_i^{\mathcal{M}}, T_i^{\mathcal{R}}) = (\frac{\sum_{j=1}^{|J_i^{\mathcal{M}}|} t_{i,j}^{\mathcal{M}}}{|\mathcal{S}^{\mathcal{M}}|}, \frac{\sum_{j=1}^{|J_i^{\mathcal{R}}|} t_{i,j}^{\mathcal{R}}}{|\mathcal{S}^{\mathcal{R}}|})$ for each job J_i. Let's consider an arbitrary pair of jobs (e.g.,J_j, J_k) with different orders between ϕ (e.g. $J_j \rightarrow J_k$ in ϕ) and $\phi^{'}$ (e.g. $J_k \rightarrow J_j$ in $\phi^{'}$). It can be noted that there must be a tail for J_j in the generalized case, as out of order condition. After we exchange their order (i.e. to be $J_k \rightarrow J_j$), we have $\breve{c}_k^{\mathcal{M}} < c_k^{\mathcal{M}}$, based on the facts that: (a). the map tasks of J_k in the simplified case will begin earlier than in the generalized case; (b). the execution time for the map tasks of J_k in the simplified case will be $\frac{\sum_{j=1}^{|J_k^{\mathcal{R}}|} t_{k,j}^{\mathcal{R}}}{|\mathcal{S}^{\mathcal{M}}|}$, which is smaller than that in the generalized case. Moreover. we have $\breve{c}_j^{\mathcal{M}} < c_j^{\mathcal{M}}$ by considering that the tail available at the last map *wave* of J_j in the generalized case. All of these indicate that, each job J_i in the simplified case will complete its map tasks earlier than (or the same as) that in the generalized case. We therefore can guarantee $\breve{c}_i^{\mathcal{R}} \geqslant c_i^{\mathcal{R}}$, for which the reason is the same as what we explained in case (1). Therefore, we have $C_{max} \geqslant \breve{C}_{max} \geqslant \breve{C}^*_{max}$.

By combining (1) and (2), we therefore have $C_{max} \geqslant \breve{C}^*_{max}$, where $\breve{C}^*_{max}$ can be estimated by using Formula (3) of *Main File* regarding $\breve{\phi}^*$.

(ii). $C_{max} \leqslant \widehat{C}^*_{max}$.

Proof: We form $\widehat{\phi}^*$ by reversing the result (denoted by ϕ_1) of *MK_JR*. In terms of Lemma 2, we have $C_{max} \leqslant \widehat{C}_{max} = \max_{1\leqslant k\leqslant n}\{\sum_{i=1}^{k} T_i^{\mathcal{M}} + \max_{1\leqslant i\leqslant k}\{\widehat{t}_i^{\mathcal{M}}\} + \sum_{i=k}^{n} T_i^{\mathcal{R}} + \max_{1\leqslant i\leqslant n}\{\widehat{t}_i^{\mathcal{R}}\}\} \leqslant \max_{1\leqslant k\leqslant n}\{\sum_{i=1}^{k} T_i^{\mathcal{M}} + \sum_{i=k}^{n} T_i^{\mathcal{R}}\} + \max_{1\leqslant i\leqslant n}\{\widehat{t}_i^{\mathcal{M}}\} + \max_{1\leqslant i\leqslant n}\{\widehat{t}_i^{\mathcal{R}}\}$. Note that $\max_{1\leqslant k\leqslant n}\{\sum_{i=1}^{k} T_i^{\mathcal{M}} + \sum_{i=k}^{n} T_i^{\mathcal{R}}\} = \widehat{C}^*_{max}$ is just the makespan estimation (Formula 3) for $\widehat{\phi}^*$ in the simplified case. According to the *Johnson's Rule*, we know that ϕ_1 is the optimal job order for makespan in the simplified case. Moreover, in terms of Theorem 2, we can obtain that $\widehat{\phi}^*$ is the worst-case job order for makespan in the simplified case. Thus we have $\widehat{C}^*_{max}$ is maximum (i.e., $C_{max} \leqslant \widehat{C}^*_{max}$) for any arbitrary job order ϕ.

Finally, our proof of Lemma 3 completes based on (i) and (ii). □

Appendix E
Proof of Theorem 1

Theorem 1. MK_JR *is an $(1+\delta)$-approximation algorithm for makespan optimization in the generalized case, where $\delta = \frac{\max_{1\leqslant i\leqslant n}\{\widehat{t}_i^{\mathcal{M}}\} + \max_{1\leqslant i\leqslant n}\{\widehat{t}_i^{\mathcal{R}}\}}{\max_{1\leqslant k\leqslant n}\{\sum_{i=1}^{k} T_i^{\mathcal{M}} + \sum_{i=k}^{n} T_i^{\mathcal{R}}\}}$, $(0 \leqslant \delta \leqslant 1)$.*

Proof: Based on Lemma 2, it holds for the job order ϕ_1 produced by *MK_JR* that,

$$\begin{aligned} C_{max} \leqslant \widehat{C}_{max} &= \max_{1\leqslant k\leqslant n}\Big\{\sum_{i=1}^{k} T_i^{\mathcal{M}} + \max_{1\leqslant i\leqslant k}\{\widehat{t}_i^{\mathcal{M}}\} + \sum_{i=k}^{n} T_i^{\mathcal{R}} + \max_{1\leqslant i\leqslant n}\{t_i^{\mathcal{R}}\}\Big\} \\ &\leqslant \max_{1\leqslant k\leqslant n}\Big\{\sum_{i=1}^{k} T_i^{\mathcal{M}} + \sum_{i=k}^{n} T_i^{\mathcal{R}}\Big\} + \max_{1\leqslant i\leqslant n}\{\widehat{t}_i^{\mathcal{M}}\} + \max_{1\leqslant i\leqslant n}\{t_i^{\mathcal{R}}\} \\ &= \breve{C}_{max} + \max_{1\leqslant i\leqslant n}\{\widehat{t}_i^{\mathcal{M}}\} + \max_{1\leqslant i\leqslant n}\{\widehat{t}_i^{\mathcal{R}}\}. \end{aligned} \quad (8)$$

Let's define $\delta = \frac{\max_{1\leqslant i\leqslant n}\{\hat{t}_i^{\mathcal{M}}\}+\max_{1\leqslant i\leqslant n}\{\hat{t}_i^{\mathcal{R}}\}}{\check{C}_{max}}, (0 \leqslant \delta \leqslant 1)$. Then Formula (8) is equivalent to

$$C_{max} \leqslant (1+\delta)\cdot \check{C}_{max}. \tag{9}$$

Note that it holds $\check{C}_{max} = \check{C}^{*}_{max}$ for *MK_JR* based on *Johnson's Rule*. Moreover, according to Lemma 3, we have $\check{C}^{*}_{max} \leqslant C^{opt}_{max}$. Therefore, we have

$$C_{max} \leqslant (1+\delta)\cdot C^{opt}_{max}. \tag{10}$$

□

Appendix F Proof of Theorem 3

Theorem 3. *Given a homogeneous environment where the Hadoop configurations of slave nodes are identical, the job orders ϕ_1 produced by* MK_JR *and ϕ_2 produced by* MK_TCT_JR *for a batch of jobs are independent of the number of slave nodes, but rather depend on the number of map/reduce slots configured within a slave node.*

Proof: Let N denote the number of slave nodes. Let $|s^{\mathcal{M}}|$ and $|s^{\mathcal{R}}|$ denote the number of map slots and reduce slots configured per slave node respectively. We thereby have $|\mathcal{S}^{\mathcal{M}}| = N\cdot|s^{\mathcal{M}}|$ and $|\mathcal{S}^{\mathcal{R}}| = N\cdot|s^{\mathcal{R}}|$. Moreover, for each job J_i, we have $T_i^{\mathcal{M}} = \frac{\sum_{j=1}^{|J_i^{\mathcal{M}}|} t_{i,j}^{\mathcal{M}}}{|\mathcal{S}^{\mathcal{M}}|} = \frac{1}{N}\cdot\frac{\sum_{j=1}^{|J_i^{\mathcal{M}}|} t_{i,j}^{\mathcal{M}}}{|s^{\mathcal{M}}|}$ and $T_i^{\mathcal{R}} = \frac{\sum_{j=1}^{|J_i^{\mathcal{R}}|} t_{i,j}^{\mathcal{R}}}{|\mathcal{S}^{\mathcal{R}}|} = \frac{1}{N}\cdot\frac{\sum_{j=1}^{|J_i^{\mathcal{R}}|} t_{i,j}^{\mathcal{R}}}{|s^{\mathcal{R}}|}$. Our proof is equivalent to validating that the rule steps of each algorithm are N-independent.

(i). ϕ_1 is N-independent for *MK_JR*.

Proof: For *MK_JR*, we only need to check its Step 2(a) and 2(b).

(1). *Step 2(a) N-independent Validation*. For each Job $J_i \in J$ in J_A, the condition $T_i^{\mathcal{M}} \leqslant T_i^{\mathcal{R}} \Leftrightarrow \frac{\sum_{j=1}^{|J_i^{\mathcal{M}}|} t_{i,j}^{\mathcal{M}}}{|\mathcal{S}^{\mathcal{M}}|} \leqslant \frac{\sum_{j=1}^{|J_i^{\mathcal{R}}|} t_{i,j}^{\mathcal{R}}}{|\mathcal{S}^{\mathcal{R}}|} \Leftrightarrow \frac{\sum_{j=1}^{|J_i^{\mathcal{M}}|} t_{i,j}^{\mathcal{M}}}{|s^{\mathcal{M}}|} \leqslant \frac{\sum_{j=1}^{|J_i^{\mathcal{R}}|} t_{i,j}^{\mathcal{R}}}{|s^{\mathcal{R}}|}$. Similarly, for each Job $J_i \in J$ in J_B, we have $T_i^{\mathcal{M}} > T_i^{\mathcal{R}} \Leftrightarrow \frac{\sum_{j=1}^{|J_i^{\mathcal{M}}|} t_{i,j}^{\mathcal{M}}}{|s^{\mathcal{M}}|} > \frac{\sum_{j=1}^{|J_i^{\mathcal{R}}|} t_{i,j}^{\mathcal{R}}}{|s^{\mathcal{R}}|}$.

(2). *Step 2(b) N-independent Validation*. For each Job J_i in the sorted set J_A, it holds $T_i^{\mathcal{M}} \leqslant T_{i+1}^{\mathcal{M}} \Leftrightarrow \frac{\sum_{j=1}^{|J_i^{\mathcal{M}}|} t_{i,j}^{\mathcal{M}}}{|\mathcal{S}^{\mathcal{M}}|} \leqslant \frac{\sum_{j=1}^{|J_{i+1}^{\mathcal{M}}|} t_{i+1,j}^{\mathcal{M}}}{|\mathcal{S}^{\mathcal{M}}|} \Leftrightarrow \frac{\sum_{j=1}^{|J_i^{\mathcal{M}}|} t_{i,j}^{\mathcal{M}}}{|s^{\mathcal{M}}|} \leqslant \frac{\sum_{j=1}^{|J_{i+1}^{\mathcal{M}}|} t_{i+1,j}^{\mathcal{M}}}{|s^{\mathcal{M}}|}$. Moreover, for each Job J_i in the sorted set J_B, it has $T_i^{\mathcal{R}} \geqslant T_{i+1}^{\mathcal{R}} \Leftrightarrow \frac{\sum_{j=1}^{|J_i^{\mathcal{R}}|} t_{i,j}^{\mathcal{R}}}{|\mathcal{S}^{\mathcal{R}}|} \geqslant \frac{\sum_{j=1}^{|J_{i+1}^{\mathcal{R}}|} t_{i+1,j}^{\mathcal{R}}}{|\mathcal{S}^{\mathcal{R}}|}\cdot t_{i+1}^{\mathcal{R}} \Leftrightarrow \frac{\sum_{j=1}^{|J_i^{\mathcal{R}}|} t_{i,j}^{\mathcal{R}}}{|s^{\mathcal{R}}|} \geqslant \frac{\sum_{j=1}^{|J_{i+1}^{\mathcal{R}}|} t_{i+1,j}^{\mathcal{R}}}{|s^{\mathcal{R}}|}$.

(ii). ϕ_2 is N-independent for *MK_TCT_JR*.

Proof: Note in Step 1 of *MK_TCT_JR*, $T_i = \frac{\sum_{j=1}^{|J_i^{\mathcal{M}}|} t_{i,j}^{\mathcal{M}}}{|\mathcal{S}^{\mathcal{M}}|} + \frac{\sum_{j=1}^{|J_i^{\mathcal{R}}|} t_{i,j}^{\mathcal{R}}}{|\mathcal{S}^{\mathcal{R}}|} = \frac{1}{N}\cdot(\frac{\sum_{j=1}^{|J_i^{\mathcal{M}}|} t_{i,j}^{\mathcal{M}}}{|s^{\mathcal{M}}|} + \frac{\sum_{j=1}^{|J_i^{\mathcal{R}}|} t_{i,j}^{\mathcal{R}}}{|s^{\mathcal{R}}|})$. Let $T_i^{'} = \frac{\sum_{j=1}^{|J_i^{\mathcal{M}}|} t_{i,j}^{\mathcal{M}}}{|s^{\mathcal{M}}|} + \frac{\sum_{j=1}^{|J_i^{\mathcal{R}}|} t_{i,j}^{\mathcal{R}}}{|s^{\mathcal{R}}|}$, then it holds $T_i = \frac{1}{N}\cdot T_i^{'}$. For *MK_TCT_JR*, we need to exam its rule steps 2, 3 as follows:

(1). *Step 2 N-independent Validation*. Note $T = (\prod_{1\leqslant i\leqslant n} T_i)^{\frac{1}{n}} = \frac{1}{N}\cdot(\prod_{1\leqslant i\leqslant n} T_i^{'})^{\frac{1}{n}}$. For each Job $J_i \in J$ in $J_A^{'}$, the condition $T_i \leqslant T \Leftrightarrow \frac{1}{N}\cdot\frac{\sum_{j=1}^{|J_i^{\mathcal{M}}|} t_{i,j}^{\mathcal{M}}}{|s^{\mathcal{M}}|} \leqslant \frac{1}{N}\cdot(\prod_{1\leqslant i\leqslant n} T_i^{'})^{\frac{1}{n}} \Leftrightarrow \frac{\sum_{j=1}^{|J_i^{\mathcal{M}}|} t_{i,j}^{\mathcal{M}}}{|s^{\mathcal{M}}|} \leqslant (\prod_{1\leqslant i\leqslant n} T_i^{'})^{\frac{1}{n}}$. Similarly, for each Job $J_i \in J$ in $J_B^{'}$, we have $T_i > T \Leftrightarrow \frac{\sum_{j=1}^{|J_i^{\mathcal{M}}|} t_{i,j}^{\mathcal{M}}}{|s^{\mathcal{M}}|} > (\prod_{1\leqslant i\leqslant n} T_i^{'})^{\frac{1}{n}}$.

(2). *Step 3 N-independent Validation*. Based on **(i)** that *MK_JR* is N-independent, therefore it holds that Step 3 is N-independent.

In summary, our proof completes in terms of (i) and (ii). □

Appendix G Proof of Lemma 4

Lemma 4. *Let ρ be the ratio of map slots to reduce slots, i.e., $\rho = \frac{|\mathcal{S}^{\mathcal{M}}|}{|\mathcal{S}^{\mathcal{R}}|}$. The optimal configuration of ρ in the simplified case for* makespan C_{max} *and* total completion time C_{tct} *are all independent of the total number of slots $|\mathcal{S}|(|\mathcal{S}| = |\mathcal{S}^{\mathcal{M}}| + |\mathcal{S}^{\mathcal{R}}|)$, but rather depends on the MapReduce workload as well as its job submission order ϕ.*

Proof: The proof is based on the formula transformation of the makespan and total completion time. Let $\rho = \frac{|\mathcal{S}^{\mathcal{M}}|}{|\mathcal{S}^{\mathcal{R}}|}$. We can transform the Formula (3) and (4) of *Main File* in the simplified case for makespan and total completion time respectively, as follows:

$$\begin{aligned}C_{max} &= \max_{1\leqslant k\leqslant n}\{\sum_{i=1}^{k} T_i^{\mathcal{M}} + \sum_{i=k}^{n} T_i^{\mathcal{R}}\}\\ &= \max_{1\leqslant k\leqslant n}\{\frac{1}{|\mathcal{S}^{\mathcal{M}}|}\sum_{i=1}^{k}\sum_{j=1}^{|J_i^{\mathcal{M}}|} t_{i,j}^{\mathcal{M}} + \frac{1}{|\mathcal{S}^{\mathcal{R}}|}\sum_{i=k}^{n}\sum_{j=1}^{|J_i^{\mathcal{R}}|} t_{i,j}^{\mathcal{R}}\}\\ &= \frac{1}{|\mathcal{S}|}\max_{1\leqslant k\leqslant n}\{(1+\rho)\sum_{i=1}^{k}\sum_{j=1}^{|J_i^{\mathcal{M}}|} t_{i,j}^{\mathcal{M}} + (1+\frac{1}{\rho})\sum_{i=k}^{n}\sum_{j=1}^{|J_i^{\mathcal{R}}|} t_{i,j}^{\mathcal{R}}\}. \end{aligned}\tag{11}$$

$$\begin{aligned}C_{tct} &= \sum_{u=1}^{n}\max_{1\leqslant k\leqslant u}\{\sum_{i=1}^{k} T_i^{\mathcal{M}} + \sum_{i=k}^{u} T_i^{\mathcal{R}}\}\\ &= \sum_{u=1}^{n}\max_{1\leqslant k\leqslant u}\{\frac{1}{|\mathcal{S}^{\mathcal{M}}|}\sum_{i=1}^{k}\sum_{j=1}^{|J_i^{\mathcal{M}}|} t_{i,j}^{\mathcal{M}} + \frac{1}{|\mathcal{S}^{\mathcal{R}}|}\sum_{i=k}^{u}\sum_{j=1}^{|J_i^{\mathcal{R}}|} t_{i,j}^{\mathcal{R}}\}\\ &= \frac{1}{|\mathcal{S}|}\sum_{u=1}^{n}\max_{1\leqslant k\leqslant u}\{(1+\rho)\sum_{i=1}^{k}\sum_{j=1}^{|J_i^{\mathcal{M}}|} t_{i,j}^{\mathcal{M}} + (1+\frac{1}{\rho})\sum_{i=k}^{u}\sum_{j=1}^{|J_i^{\mathcal{R}}|} t_{i,j}^{\mathcal{R}}\}. \end{aligned}\tag{12}$$

Formula (G.1), (G.2) show that the optimal configuration of ρ for makespan and total completion time in the simplified case is independent of the total number of slots $|\mathcal{S}|$, but rather depends on the MapReduce workload and job submission order. Hence, Lemma 4 holds.

□

Appendix H Lower and Upper Bound Makespan Evaluation

Note that Lemma 2 gives a formula to compute the upper bound makespan for a MapReduce workload under an

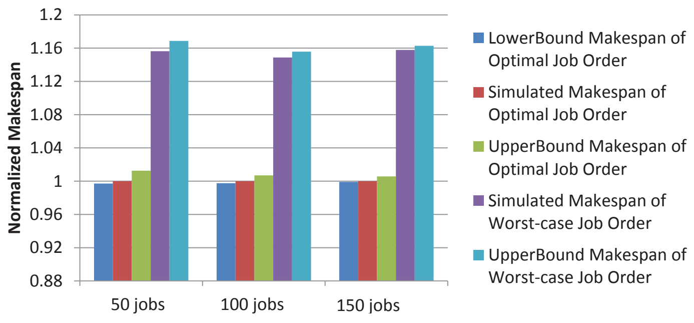


Fig. 1: The normalized comparison result of simulated makespan, lower bound makespan and upper bound makespan for optimal job order produced by *MK_JR*, as well as simulated makespan and upper bound makespan for worst-case job order based on Theorem C. Particularly, the results are normalized by dividing with simulated makespan of the optimal job order.

arbitrary job submission order. Moreover, Lemma 3 tell us how to derive a lower bound makespan as well as a upper bound makespan for all job submission orders. We perform experiments to evaluate the tightness of these lower (or upper) bound makespan with synthetic Facebook workloads. As shown in Figure 1, we compute the lower bound and upper bound makespan of the optimal job order produced by *MK_JR*, according to Lemma 2 and Lemma 3. In contrast, the upper bound makespan of the worst-case job order can be computed based on Theorem 2 and Lemma 2. Moreover, we derive the simulated makespan for optimal job order as well as worst-case job order with *MREstimator* to evaluate the tightness of lower and upper bound makespan. All of these results are normalized by dividing with simulated makespan of the optimal job order. The experimental results show that, both the lower bound makespan and upper bound makespan are very close to the corresponding simulated makespan, which validates the tightness of the upper bound and lower bound makespan models.

Appendix I
Accuracy and Efficiency Validation of *PCP*

Recall in Section 6.3 of the *Main File*, we provide a method *PCP* to address the possible time efficiency problem for proposed slot enumerating algorithms (e.g., Algorithm MK_SF_JR, Algorithm MK_TCT_SF_JR) in map/reduce slot configuration optimization when the total number of slots is very large. This section aims to validate the accuracy and time efficiency of *PCP*.

As shown in Table 1, we make a contrast experiment for *PCP* by comparing its predicted results (rows labeled with '*PCP*') with those ones (rows labeled with '*algorithm*') produced by corresponding map/reduce slot configuration algorithms for a Facebook workload of 100 jobs. Particularly, in our experiment, we choose a total number of 100 slots as a basis for *PCP* to deduce the map/reduce slot configuration results when the total number of slots is very large (e.g., 200, 400, 800, 1600). That is, we first run each slot configuration algorithm to obtain the corresponding optimized result for a small-size number of 100 slots. Then we use it to deduce the corresponding result for a large number of slots, according to *PCP*. We have the following observations:

First, for each size of total slots (e.g., 100, 200 ,400, 800, 1600), both the results of makespan (MK) and total completion time (TCT) deduced by *PCP* are much close to the ones obtained by running slot configuration algorithms under that given size. For example, there is only a bit difference of $3(\approx 211.34 - 207.98)$ seconds for MK and $6(\approx 18175.03 - 18169.31)$ seconds for TCT, between the results from '*PCP*' and '*algorithm*' for a total number of 1600 slots, with respect to Algorithm MK_SF_JR. It validates the accuracy of **PCP** approach.

Second, there is a significant Execution Time (ET) reduction for the proposed algorithms combined with the *PCP* method when the slot size is large. We can witness that, the execution time of algorithm becomes almost twice when we double the size of slots. For example, for Algorithm MK_SF_JR, when the size of total slots is 400, it takes 10.3 seconds to get the optimized result. However, it consumes 21 seconds when we double the size of total slots to be 800. In comparison, with *PCP*, we only need to use 2.48 seconds, reducing the computation time significantly.

In summary, the *PCP* approach can reduce the computation time significantly for proposed algorithms while keeping the high accuracy results (i.e., makespan and total completion time) when the size of total slots is very large.

Appendix J
Accuracy Validation for MREstimator

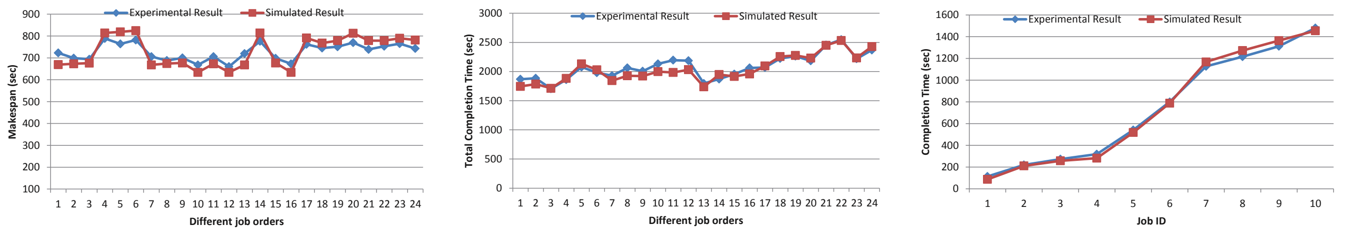
We validate the accuracy of our *MREstimator* by comparing the simulated results of testbed workloads with the experimental results. Our validation work consists of two flavors. First, we consider the makespan as well as total completion time for all possible job orders of a testbed workload. Figure 2(a) and Figure 2(b) present the results for all 24 job orders of a batch of 4 jobs. We can note that the calculation results of both makespan and total completion time are very close (errors within 8%) to the experimental results. On the other hand, we focus on a single job order of a testbed workload and consider the completion time for each job. Figure 2(c) shows completion time for each job of a testbed workload of 10 jobs under the job order produced by *MK_TCT_JR*. We can observe that the curve of simulated results almost overlaps (errors within 5%) with that of experiment results. In summary, all of these forcefully validate the accuracy of our *MREstimator*.

Appendix K
Performance Impact of Inaccuracy in Estimating Map/Reduce Task Time

In previous sections, we take the average task execution time for map/reduce tasks of each job as input to optimize the performance of a workload. However, in practice, the execution time of map/reduce tasks can be fluctuated up and

Total Slots	Result Source	Algorithm MK_SF_JR				Algorithm MK_TCT_SF_JR			
		Slot Ratio ($\frac{mapSlots}{TotalSlots}$)	MK (sec)	TCT (sec)	ET (sec)	Slot Ratio ($\frac{mapSlots}{TotalSlots}$)	MK (sec)	TCT (sec)	ET (sec)
100	*PCP*	$\frac{73}{100} = 0.73$	3075.58	266758.98	2.48	$\frac{72}{100} = 0.72$	3129.10	68329.93	4.33
	algorithm	$\frac{73}{100} = 0.73$	3075.58	266758.98	2.48	$\frac{72}{100} = 0.72$	3129.10	68329.93	4.33
200	*PCP*	$\frac{146}{200} = 0.73$	1545.47	134160.39	2.48	$\frac{144}{200} = 0.72$	1567.91	34359.75	4.33
	algorithm	$\frac{147}{200} = 0.735$	1535.18	135302.15	5.08	$\frac{144}{200} = 0.72$	1567.91	34359.75	8.88
400	*PCP*	$\frac{292}{400} = 0.73$	781.78	67904.30	2.48	$\frac{288}{400} = 0.72$	814.30	15515.18	4.33
	algorithm	$\frac{294}{400} = 0.735$	775.76	68336.39	10.30	$\frac{286}{400} = 0.715$	804.66	15518.56	18.51
800	*PCP*	$\frac{584}{800} = 0.73$	400.57	34666.10	2.48	$\frac{576}{800} = 0.72$	413.20	8819.44	4.33
	algorithm	$\frac{588}{800} = 0.735$	396.13	34703.35	21.0	$\frac{568}{800} = 0.71$	409.70	8818.06	37.61
1600	*PCP*	$\frac{1168}{1600} = 0.73$	211.34	18169.31	2.48	$\frac{1152}{1600} = 0.72$	219.86	6277.49	4.33
	algorithm	$\frac{1176}{1600} = 0.735$	207.98	18175.03	42.06	$\frac{1164}{1600} = 0.727$	218.03	6235.14	73.45

TABLE 1: The comparison of map/reduce slot configuration (Slot Ratio), makespan (MK) and total completion time (TCT) for slot configuration optimization algorithms under two different result sources: *PCP* and *algorithm* for the Facebook workload of 100 jobs. The '*PCP*' represents that the map/reduce slot configuration as well as optimized job submission order for the case of current (large) size of total slots is figured out based on the result from a small-size number of total slots (e.g., 100 slots in total in our example), in terms of PCP. In contrast, the '*algorithm*' denotes that the results of slot configuration and job submission order are obtained by running slot configuration algorithms with given size of total slots (e.g., 200, 400, 800, 1600). *ET* denotes the execution time needed for the corresponding optimization algorithm.



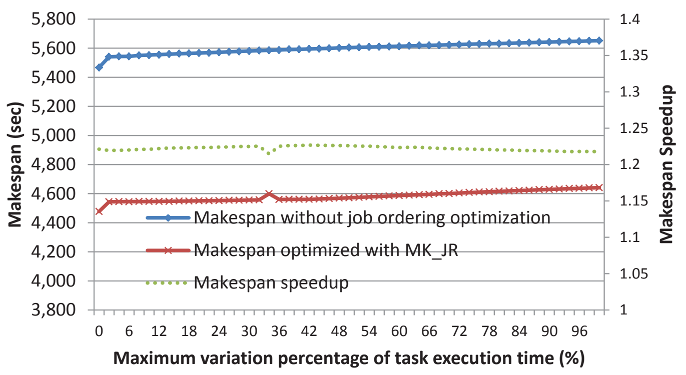
(a) *Makespan* for a batch of 4 jobs in all job orders (b) *Total completion time* for a batch of 4 jobs in all job orders (c) The *completion time* for each job of a testbed workload of 10 jobs under a job order

Fig. 2: Simulation results versus experimental results for testbed workloads.

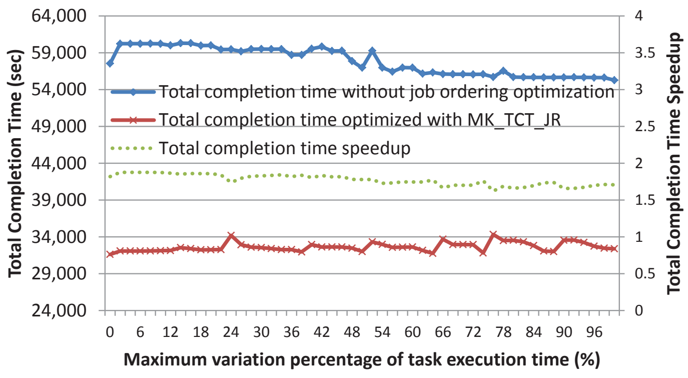
down the average value, depending on specific applications. In this section, we study the impact of such a variation on the overall performance of a workload for our job ordering algorithms.

In our experiment below, we assume that it follows the uniform distribution for the variation percentage of map/reduce task execution time relative to the corresponding average task execution time. We take the testbed workload of 100 jobs and the synthetic Facebook workload of 100 jobs as examples to study the influence of different task execution time variations on makespan and total completion time, as the results shown in Figure 3(a) and Figure 3(b), respectively. The x-axis gives the maximum variation percentage (i.e., variation domain) of map/reduce task execution time relative to its average value. We generate the specific execution time for each map/reduce task with the monte carlo method that follows the uniform probability distribution for the variation percentage of execution time subject to the given variation domain. For example, Point 12 at the x-axis in Figure 3(a) means that the execution time for each map/reduce task fluctuates up or down randomly within 12% relative to the average task execution time with uniform probability distribution. Hence, Point 0 at the x-axis represents the performance results under the pure average task execution time. Moreover, we normalize the makespan (or total completion time) with makespan speedup (or total completion time speedup) through dividing the makespan (or total completion time) of originally unoptimized job order by the one of optimized one with our job ordering algorithms (*MK_JR* and *MK_TCT_JR*). Particularly, it is worth mention that the job ordering optimization performed is still based on the average task execution time.

The results of Figure 3 show that there is a small increase for makespan (and total completion time) for both cases of unoptimized job order and optimized one with our proposed job ordering algorithms under different degrees of variation in map/reduce task execution time, whereas the relative performance improvement (i.e., makespan speedup, total completion time speedup) is fine or even better under this case. It indicates that we can take the average task execution time as input for job ordering algorithms under the case of inaccuracy in estimating map/reduce task time, assuming that the variation percentage of task execution time follows uniform probability distribution.



(a) *Makespan* for the testbed workload of 100 jobs.



(b) *Total completion time* for the synthetic Facebook workload of 100 jobs.

Fig. 3: The performance impact of map/reduce task execution time variation for different variation percentages.