# Exploiting Large System Dynamics for Designing Simple Data Center Schedulers

Yousi Zheng*, Ness B. Shroff*‡, R. Srikant†, Prasun Sinha‡
*Dept. of ECE, The Ohio State University, Columbus, OH 43210
‡Dept. of CSE, The Ohio State University, Columbus, OH 43210
†Dept. of ECE, University of Illinois at Urbana-Champaign, Urbana, IL 61801

***Abstract*—The number and size of data centers has seen a rapid growth in the last few years. It is no longer uncommon to see large data centers with thousands or even tens of thousands of machines. Hence, it is critical to develop scalable scheduling mechanisms for processing the enormous number of jobs handled by popular paradigms such as the MapReduce framework. This work explores the possibility of simplifying the scheduling procedure by exploiting the "largeness" of the data center system. Specifically, we consider the problem of minimizing the total flow time of a sequence of jobs under the MapReduce framework, where the jobs arrive over time and need to be processed through both Map and Reduce procedures before leaving the system. We show that any work-conserving scheduler is asymptotically optimal under a wide range of traffic loads, including the heavy traffic limit. Our results are shown for scenarios in which the tasks can be preempted and served in parallel over different machines, as well as scenarios when each task has to be served only on one machine and cannot be preempted. This result implies, somewhat surprisingly, that when we have a large number of machines, there is little to be gained by optimizing beyond ensuring that a scheduler should be work-conserving. For long running applications, we also study the relationship between the number of machines and total running time, and show sufficient conditions to guarantee the asymptotic optimality of work-conserving schedulers. Further, we run extensive simulations, that indeed verify that when the total number of machines is large, state-of-the-art work-conserving schedulers have similar and close-to-optimal delay performance.**

## I. Introduction

The number of large-scale data centers (e.g., those with tens of thousands of machines) is rapidly increasing. MapReduce is a framework designed to process massive amounts of data in data centers [1]. Although it was first proposed by Google [1], today, many other companies including Microsoft, Yahoo, and Facebook also use this framework. This framework is being widely used for applications such as search indexing, distributed searching, web statistics generation, and data mining.

In MapReduce, each arriving job consists of a set of Map tasks and Reduce tasks. The *scheduler* is centralized and responsible for making decisions on which task will be executed at what time and on which machine. The key metric considered in this paper is the *total delay in the system* per job, which is the time it takes a job, since its arrival into the system, to be processed fully. This includes both the delay in waiting before the first task in the job begins to be processed, and the time for processing all tasks in the job.

A critical consideration for the design of the scheduler is the *dependence* between the Map and Reduce tasks. For each job, the Map tasks need to be finished before starting any of its Reduce tasks[1] [1], [3]. Various scheduling solutions have been proposed within the MapReduce framework [2]–[6]. However, under some weak assumptions it was proven [7] that there is no upper bound on the competitive ratio of any causal scheduler for total delay (i.e. flow time, which is the time between the job arrival and the completion of all the phases in the job) minimization problem. In an attempt to minimize the total delay, a new metric to analyze the performance of schedulers called *efficiency ratio* was introduced in [7], [8], where loose bounds were provided on the performance of general work-conserving schedulers, and under the special case of preemptive and paralleizable tasks, it was shown that a scheduler called ASRPT can guarantee an efficiency ratio of two. However, ASPRT requires a sorting operation whose complexity grows rapidly with the number of machines in the data center. What we show in this paper is that the scheduler design can be significantly simplified by taking advantage of scale. Moreover, this scale also allows us to design simple schedulers that not only provide a bounded efficiency ratio as in [7], [8], but in fact approach optimality (i.e., the efficiency ratio goes to 1) as the number of machines grows large. Thus, by exploiting the scale inherent in current data centers, we can improve both the complexity of scheduler design in MapReduce-style multiple-phases system and the performance guarantees that we can provide.

We briefly summarize the main contributions of our work:

- Under certain weak assumptions, we show that for the flow-time minimization problem any work-conserving scheduler is asymptotically optimal in two classes of scenarios. (i) Preemptive and Parallelizeable: in which the Reduce tasks can be preempted and served in parallel over different machines (Section III). (ii) Non-Preemptive and Non-Parallelizeable: when each Reduce task has to be served only on one machine and cannot be preempted (Section IV). These scenarios can be thought to capture two extreme cases of *data locality*, the first when the cost of data migration is negligible, and the second when the

This work is supported in part by a MURI grant from the Army Research Office W911NF-12-1-0385.

[1]Here, we consider the most popular case in reality without the Shuffle phase. For discussion about the Shuffle phase, see [2].

cost of data migration is prohibitive.

- For long running applications, we study the relationship between the number of machines $N$ and total running time $T$. We provide sufficient conditions to guarantee the asymptotic optimality of work-conserving schedulers, when $T$ is function of $N$ (Section V).
- We verify via simulations (Section VI) that while state-of-the-art work-conserving schedulers have different delay performance for a small number of machines $N$, these differences rapidly vanish as $N$ becomes large, and the performance is virtually indistinguishable and close to optimal for a few hundred machines.

Our results provide the following two surprising properties: First, in a large system, it is not necessary to implement complex schedulers, as long as they honor the work-conserving principle, thus ensuring both high performance and scalability. Second, under appropriate and general assumptions, work-conserving schedulers can guarantee asymptotic optimality under both the *Noah Effect* [9] (a large amount of workload arrives into the system in the preemptive and parallelizable scenario) and *Joseph Effect* [9] (a large number of cumulative running jobs remain in the system in the non-preemptive and non-parallelizable scenario).

## II. System Model and Asymptotic Optimality

### A. System Model under MapReduce Framework

MapReduce has two elemental processes: Map and Reduce. For the Map tasks, the inputs are divided into several small sets, and processed by different machines in parallel. The Reduce tasks then operate on the intermediate data, possibly running the operation on multiple machines in parallel to generate the final result. Note that for a given job, the Reduce tasks have to start executing after all (or a subset) of Map tasks for that job have been completed.

Consider a data center with $N$ machines and $n$ jobs that arrive during a time period $T$. We assume that each machine can only process one job at a time. A machine could represent a processor, a core in a multi-core processor or a virtual machine. We assume the scheduler periodically collects the information on the state of jobs running on the machines, which is used to make scheduling decisions.

We assume that each job $i$ brings $M_i$ units of workload for its Map tasks and $R_i$ units of workload for its Reduce tasks. Here, the $R_i$ workload belongs to Reduce operations. Each Map task has 1 unit of workload[2], however, each Reduce task can have multiple units of workload. Time is slotted and each machine can run one unit of workload in each time slot.

We assume that the number of job arrivals in each time slot is *i.i.d.*, and the arrival rate is $\lambda$. Assume that $\{M_i\}$ are *i.i.d.* with expectation $\overline{M}$, and $\{R_i\}$ are *i.i.d.* with expectation $\overline{R}$. In time slot $t$ for job $i$, $m_{i,t}$ and $r_{i,t}$ machines are scheduled for Map and Reduce tasks, respectively. As we know, each job contains several tasks. We assume that job $i$ contains $K$ tasks, and the workload of the Reduce task $k$ of job $i$ is $R_{i,k}$. Thus, for any job $i$, $\sum_{k=1}^{K} R_{i,k} = R_i$. In time slot $t$ for job $i$, $r_{i,t}^{(k)}$ machines are scheduled for the Reduce task $k$. As each Reduce task may consist of multiple units of workload, it can be processed in either preemptive and parallelizable or non-preemptive and non-parallelizable fashion based on the type of scheduler [7].

**Definition 1.** *A scheduler is called **preemptive and parallelizable** if Reduce tasks belonging to the same job can run in parallel on multiple machines, can be interrupted by other tasks, and can be rescheduled to different machines in different time slots.*

*A scheduler is called **non-preemptive and non-parallelizable** if each Reduce task can only be scheduled on one machine and, once started, it must keep running without any interruption.*

The schedulers are classified into these two different scenarios based on implementation consideration. In data centers, there is additional cost of data migration and initialization for each job. Such a cost could be caused by limited I/O speed, limited network transmission rate, and large amount of initialization work needed before a job. When the cost of data migration and initialization is not significant, then the schedulers can interrupt and migrate data from one machine to another to avoid unnecessary delay, which corresponds to the preemptive and parallelizable scenario. Alternatively, when such a cost is large, then the schedulers should avoid any action of interruption and migration, which corresponds to the non-preemptive and non-parallelizable scenario. Which scenario is more appropriate depends on the application, network topology and transmission rate, and the I/O speed.

Let the arrival time of job $i$ be $a_i$, the time slot in which the last Map task finishes execution be $f_i^{(m)}$, and the time slot in which all Reduce tasks are completed be $f_i^{(r)}$. For the preemptive and parallelizable scenario, the problem definition is as follows:

$$\begin{aligned} &\min_{m_{i,t},r_{i,t}} \sum_{i=1}^{n} \left( f_i^{(r)} - a_i + 1 \right) \\ &s.t. \quad \sum_{i=1}^{n} (m_{i,t} + r_{i,t}) \le N,\ r_{i,t} \ge 0,\ m_{i,t} \ge 0,\ \forall t, \\ &\sum_{t=a_i}^{f_i^{(m)}} m_{i,t} = M_i,\ \sum_{t=f_i^{(m)}+1}^{f_i^{(r)}} r_{i,t} = R_i,\ \forall i \in \{1, ..., n\}. \end{aligned} \tag{1}$$

In the non-preemptive and non-parallelizable scenario, the optimization problem in this scenario is as in Eq. (1), with additional constraints representing the non-preemptive and non-parallelizable nature as follows:

$$r_{i,t}^{(k)} = 0 \text{ or } 1,\ r_{i,t}^{(k)} = 1 \text{ if } 0 < \sum_{s=0}^{t-1} r_{i,s}^{(k)} < R_{i,k}. \tag{2}$$

The scheduling problems (both preemptive and parallelizable and non-preemptive and non-parallelizable) are NP-hard in the strong sense [7].

[2]Because the Map tasks are independent and have small workload [5], such assumption is valid.

### B. Asymptotic Optimality of Schedulers

**Definition 2.** *For a data center with $N$ machines, let $T$ represent the total time slots. Define $F^S(N,T)$ as the total flow time of a scheduler $S$, and $F^*(N,T)$ as the minimum total flow time over all feasible schedulers. Then, a scheduler $S$ is called* ***asymptotically optimal*** *if*

$$\lim_{N\to\infty} \frac{F^S(N,T)}{F^*(N,T)} = 1 \text{ w.p.1 for any given } T. \quad (3)$$

**Definition 3.** *In a data center with $N$ machines,* ***traffic intensity*** *$\rho_N$ is the ratio of expected arrival workload over the processing ability of $N$ machines. When $\rho_N$ is a constant for any value of $N$, it is written as $\rho$ for short.*

Note that when we let $N$ goes to infinity, we also let the corresponding arrival load go to infinity, such that $\rho_N > 0$ (and also $\rho_N \to 1$), to avoid trivialities.

Also, we define work-conserving schedulers as follows.

**Definition 4.** *A scheduler is called* ***work-conserving scheduler*** *if jobs are always scheduled when there are idle machines without violating the dependencies among multiple phases.*

The definition contains two parts. First, jobs are scheduled without unnecessary waiting, i.e., there is no time slot that jobs are waiting and machines are idle. Second, the dependence between multiple phases (called **Phase Precedence** [8]) must hold. In MapReduce framework, specially, the phase dependence between Map and Reduce tasks has to always hold. That is for a job, Reduce tasks can only be processed if all (or in general a subset) of Map tasks are completed for that job. In the following sections, we will show the asymptotic optimality of the work-conserving schedulers that are the most popular and widely used in data centers.

### C. Assumptions

In this part, we list the assumptions which will be used in the following sections. It does not mean that all the assumptions are needed in each theorem. We will point out which assumption is required when we state the theorems.

- (A1) $\{M_i, i = 1,2,...\}$ are *i.i.d.* with expectation $0 < \overline{M} < \infty$, $\{R_i, i = 1,2,...\}$ are *i.i.d.* with expectation $0 < \overline{R} < \infty$.
- (A2) $\{M_i, i = 1,2,...\}$ are *i.i.d.* with expectation $0 < \overline{M} < \infty$ and variance $\sigma_m^2 < \infty$, $\{R_i, i = 1,2,...\}$ are *i.i.d.* with finite expectation $0 < \overline{R} < \infty$ and finite variance $\sigma_r^2 < \infty$.
- (A3) The workload distribution of each job does not change with $T$ and $N$.
- (A4) For each $N$, the number of arrival jobs $A_t^{(N)}$ in each time slot $t$ are *i.i.d.*, and are independent of $M_i$ and $R_i$.
- (A5) For all $t$, $A_t^{(N)}$ and $\frac{A_t^{(N)}}{N}$ converges to infinity and a constant w.p.1, respectively, as $N$ goes to infinity.
- (A6) The traffic intensity $\rho$ is a constant, where $0 < \rho < 1$.
- (A7) The traffic intensity $\rho_N \to 1$ as $N \to \infty$, and
$$\liminf_{N\to\infty} \frac{(1-\rho_N)\sqrt{N}}{\sqrt{\log\log N}} > \frac{\sqrt{2}(\sigma_m+\sigma_r)}{\sqrt{\overline{M}+\overline{R}}}. \quad (4)$$
- (A8) When the traffic intensity $\rho_N \to 1$, if $\frac{A_t^{(N)}}{N} \to \frac{1}{\overline{M}+\overline{R}}$ w.p.1 as $N \to \infty$, then
$$\liminf_{N\to\infty} \frac{\left(E\left[\frac{A_t^{(N)}}{N}\right] - \frac{A_t^{(N)}}{N}\right)\sqrt{N}}{\sqrt{\log\log N}} \geq 0 \text{ w.p.1.} \quad (5)$$
- (A9) In non-preemptive and non-parallelizable scenario, the number of Reduce tasks in each job are *i.i.d.*, and the workload of each Reduce task are also *i.i.d.*. Also, the largest number of Reduce tasks in each job are bounded.
- (A10) The traffic intensity $\rho_N \to 1$ as $N \to \infty$, and
$$\liminf_{N\to\infty} \frac{(1-\rho_N)\sqrt{N}}{\sqrt{\log\log N}} = \infty. \quad (6)$$

**Remark 1.** *Note that the assumptions of the arrival jobs are weak. In Sections III and IV, we will see that, when $\rho < 1$, we allow for heavy tailed distributions and only requires finite mean ((A1)). Similarly, for appropriate heavy traffic region of $\rho_N$ goes to 1, we allow for heavy tailed distributions and only requires finite mean and variance ((A2)).*

**Lemma 1.** *Assumption (A5) implies that $\lim_{N\to\infty} A_t^{(N)}/N = c$ w.p.1, where $c$ is a constant. By the definition of traffic intensity $\rho$, assumptions (A3) and (A4) imply that*

$$\rho = \frac{1}{N} E\left[\sum_{j=1}^{A_t^{(N)}} (\overline{M}+\overline{R})\right] = \frac{1}{N} E\left[A_t^{(N)}\right](\overline{M}+\overline{R}). \quad (7)$$

*and*

$$c \leq \lim_{N\to\infty} E\left[A_t^{(N)}\right]/N = \frac{\rho}{\overline{M}+\overline{R}}. \quad (8)$$

*Similarly, assumptions (A3) and (A9) or (A10) imply that*

$$c \leq \lim_{N\to\infty} E\left[A_t^{(N)}\right]/N = \frac{1}{\overline{M}+\overline{R}}. \quad (9)$$

**Remark 2.** *Roughly speaking, (A7) implies that*

$$1-\rho_N > \frac{\sqrt{2}(\sigma_m+\sigma_r)}{\sqrt{(\overline{M}+\overline{R})}}\sqrt{\frac{\log\log N}{N}}, \quad (10)$$

*when $N$ is large enough. Similarly, (A10) implies that $1-\rho_N \gg \frac{\sqrt{\log\log N}}{\sqrt{N}}$, when $N$ is large enough.*

*In other words, assumptions (A7) and (A10) define the appropriate heavy traffic regions, which show the convergence rate of $\rho_N$ to 1 corresponding to $N$.*

## III. Asymptotic Optimality of Work-conserving Schedulers in Preemptive and Paralleled Scenario

In this section, we will show that all the work-conserving schedulers are asymptotically optimal. For simplicity, we let $M_j^{(t)}$ and $R_j^{(t)}$ be the Map and Reduce workload of $j^{th}$ arrival job in time slot $t$, respectively, in the following sections.

**Theorem 1.** *Under assumptions (A1) and (A3)-(A6), any work-conserving scheduler is asymptotically optimal.*

*Proof:* To prove Theorem 1, we first show that if the machines only serve the Map tasks of jobs arriving in the current time slot and the Reduce tasks of jobs arrived in the previous time slot, then the probability that all these task can be served immediately goes 1 as $N$ goes to infinity. In the following part, "no unnecessary delay" or "be served immediately" means that all the available phases can be served. Obviously, the phases, whose dependent phases are running, are unavailable and cannot be served for any schedulers.

For the first time slot, there is no Reduce tasks corresponding to previous time slot. When time slot $t > 1$, there could be both new arrival workload of Map and available workload of Reduce. Thus, if we can prove no unnecessary delay for each job when $t > 1$, then it is obvious that there will be no delay when $t = 1$.

For any other time slot $t > 1$, we have

$$\begin{aligned}
&\lim_{N\to\infty} \frac{\sum_{j=1}^{A_{t-1}^{(N)}} R_j^{(t-1)} + \sum_{j=1}^{A_t^{(N)}} M_j^{(t)} - N}{N} \\
&= \lim_{A_{t-1}^{(N)}\to\infty} \frac{\sum_{j=1}^{A_{t-1}^{(N)}} R_j^{(t-1)}}{A_{t-1}^{(N)}} \lim_{N\to\infty} \frac{A_{t-1}^{(N)}}{N} + \\
&\lim_{A_t^{(N)}\to\infty} \frac{\sum_{j=1}^{A_t^{(N)}} M_j^{(t)}}{A_t^{(N)}} \lim_{N\to\infty} \frac{A_t^{(N)}}{N} - 1 \\
&= \left(\overline{R} + \overline{M}\right) c - 1 \le \rho - 1 \text{ w.p.1.}
\end{aligned} \tag{11}$$

Since $\rho < 1$, then Eq. (11) is less than 0 w.p.1. Then, for any time slot $t$, the Map tasks of new arrival jobs and the Reduce tasks of jobs which arrive in the previous time slot can be finished at time slot $t$ w.p.1, as $N$ goes to infinity.

Let $F^{wc}(N,T)$ be the total flow time of any work-conserving scheduler, and let $F^*(N,T)$ is the optimal total flow time of all feasible schedulers.

We define the set $\Xi_t$ as follows:

$$\Xi_t \triangleq \begin{cases} \left\{ \epsilon : \frac{\sum_{j=1}^{A_t^{(N)}} M_j^{(t)}}{N} \ge 1 \text{ as } N \to \infty \right\}, & t = 1 \\ \left\{ \epsilon : \frac{\sum_{j=1}^{A_{t-1}^{(N)}} R_j^{(t-1)} + \sum_{j=1}^{A_t^{(N)}} M_j^{(t)}}{N} \ge 1 \text{ as } N \to \infty \right\}, & t \ge 2 \end{cases}. \tag{12}$$

Then, set $\Xi_t$ is a null set for any $t$ by the previous proof. $\bigcup_{t=1}^{T} \Xi_t$, which is a finite union of null sets, is also a null set. Thus, $P\left(\bigcap_{t=1}^{T} (\Omega \setminus \Xi_t)\right) = P\left(\Omega \setminus \bigcup_{t=1}^{T} \Xi_t\right) = 1$, i.e., for any $T$, when $N$ goes to infinity, all the Map and Reduce workload will be finished either in the time slot they arrive or the very next time slot w.p.1. In the set $\Omega \setminus \bigcup_{t=1}^{T} \Xi_t$, any work-conserving scheduler has the same total flow time as the infinite number of machines scenario w.p.1, as $N$ goes to infinity. If there are infinite number of machines, then the total flow time is a lower bound of the finite number of machines scenario. In other words, $\left(\Omega \setminus \bigcup_{t=1}^{T} \Xi_t\right) \subseteq \left\{ \lim_{N\to\infty} \frac{F^{wc}(N,T)}{F^*(N,T)} = 1 \right\}$. Thus, $P\left( \lim_{N\to\infty} \frac{F^{wc}(N,T)}{F^*(N,T)} = 1 \right) = 1$. Eq. (3) is achieved. ■

**Corollary 1.** *In Theorem, we can also get that*

$$\lim_{T\to\infty} \lim_{N\to\infty} \frac{F^S(N,T)}{F^*(N,T)} = 1 \text{ w.p.1.} \tag{13}$$

*Proof:* Similar to the proof of Theorem 1, $\bigcup_{t=1}^{\infty} \Xi_t$, which is a countable union of null sets, is also a null set. Hence, using the same method to prove Eq. (3), we can get Eq. (13). ■

In previous results, we fixed traffic intensity $\rho$ as a constant which is less than 1. In the next theorem, we will discuss heavy traffic scenario, and show that work-conserving scheduler is still asymptotically optimal in appropriate heavy traffic regions.

**Theorem 2.** *In the MapReduce framework, assume (A2)-(A5) and (A7)-(A8) hold. Then, any work-conserving scheduler is asymptotically optimal.*

*Proof:* Similarly to the proof of Theorem 1, we focus on the workload of Map tasks which corresponds to jobs arrive in the current time slot, and the workload of Reduce tasks which corresponds to jobs arrived in the previous time slot. If such workload can be immediately served by any work-conserving scheduler, then by the same proof of Theorem 1, we can directly achieve Theorem 2.

For time slot $t > 1$, there could be both Map arrival tasks and remaining Reduce tasks. For time slot 1, there are only Map tasks are available in the framework, i.e., there is a lighter workload in time slot 1 than other time slots. Thus, we only need to focus on time slot $t > 1$. If the workload can be immediately served at $t > 1$, then it is also true when $t = 1$.

For time slot $t \ge 2$, we have

$$\begin{aligned}
&\limsup_{N\to\infty} \frac{\sum_{j=1}^{A_{t-1}^{(N)}} R_j^{(t-1)} + \sum_{j=1}^{A_t^{(N)}} M_j^{(t)} - N}{\sqrt{N \log\log N}} \\
&= \limsup_{A_{t-1}^{(N)}\to\infty} \frac{\sum_{j=1}^{A_{t-1}^{(N)}} (R_j^{(t-1)} - \overline{R})}{\sqrt{A_{t-1}^{(N)} \log\log A_{t-1}^{(N)}}} \lim_{N\to\infty} \frac{\sqrt{A_{t-1}^{(N)} \log\log A_{t-1}^{(N)}}}{\sqrt{N \log\log N}} \\
&+ \limsup_{A_t^{(N)}\to\infty} \frac{\sum_{j=1}^{A_t^{(N)}} (M_j^{(t)} - \overline{M})}{\sqrt{A_t^{(N)} \log\log A_t^{(N)}}} \lim_{N\to\infty} \frac{\sqrt{A_t^{(N)} \log\log A_t^{(N)}}}{\sqrt{N \log\log N}} \\
&- \liminf_{N\to\infty} \frac{N - A_t^{(N)}\overline{M} - A_{t-1}^{(N)}\overline{R}}{\sqrt{N \log\log N}}.
\end{aligned} \tag{14}$$

By Eq. (9) in Lemma 1, we can get that

$$\begin{aligned}&\lim_{N\to\infty}\frac{1}{\sqrt{N\log\log N}}\sqrt{A_t^{(N)}\log\log A_t^{(N)}}\\ =&\lim_{N\to\infty}\sqrt{\frac{A_t^{(N)}}{N}}\sqrt{\frac{\log(\log\frac{A_t^{(N)}}{N}+\log N)}{\log\log N}}\\ =&\sqrt{c}\leq\frac{1}{\sqrt{\overline{M}+\overline{R}}}\text{ w.p.1.}\end{aligned}\tag{15}$$

By the Law of the Iterated Logarithm (LIL) [10], we can get that

$$\limsup_{A_{t-1}^{(N)}\to\infty}\frac{\sum_{j=1}^{A_{t-1}^{(N)}}(R_j^{(t-1)}-\overline{R})}{\sqrt{A_{t-1}^{(N)}\log\log A_{t-1}^{(N)}}}=\sqrt{2}\sigma_r\text{ w.p.1}\tag{16}$$

and

$$\limsup_{A_t^{(N)}\to\infty}\frac{\sum_{j=1}^{A_t^{(N)}}(M_j^{(t)}-\overline{M})}{\sqrt{A_t^{(N)}\log\log A_t^{(N)}}}=\sqrt{2}\sigma_m\text{ w.p.1.}\tag{17}$$

Based on Eq. 8 in Lemma 1, there are two cases: $c<\frac{1}{\overline{M}+\overline{R}}$ or $c=\frac{1}{\overline{M}+\overline{R}}$.

1) Case 1: $c<\frac{1}{\overline{M}+\overline{R}}$. We get

$$\begin{aligned}&\liminf_{N\to\infty}\frac{\sqrt{N}}{\sqrt{\log\log N}}\left(1-\frac{A_t^{(N)}}{N}(\overline{M}+\overline{R})\right)\\ =&\left(\frac{1}{\overline{M}+\overline{R}}-c\right)\lim_{N\to\infty}\frac{(\overline{M}+\overline{R})\sqrt{N}}{\sqrt{\log\log N}}\text{ w.p.1.}\end{aligned}\tag{18}$$

Since $c<\frac{1}{M+R}$ and $\lim_{N\to\infty}\frac{\sqrt{N}}{\sqrt{\log\log N}}=\infty$, we can get that

$$\liminf_{N\to\infty}\frac{(1-\frac{A_t^{(N)}}{N}(\overline{M}+\overline{R}))\sqrt{N}}{\sqrt{\log\log N}}=\infty\text{ w.p.1.}\tag{19}$$

2) Case 2: $c=\frac{1}{\overline{M}+\overline{R}}$. In this case, by assumption (A8), we can get that

$$\begin{aligned}&\liminf_{N\to\infty}\frac{N-A_t^{(N)}\overline{M}-A_{t-1}^{(N)}\overline{R}}{\sqrt{N\log\log N}}=\liminf_{N\to\infty}\frac{\sqrt{N}(1-\rho_N)}{\sqrt{\log\log N}}\\ &+\liminf_{N\to\infty}\frac{\sqrt{N}(E\left[\frac{A_t^{(N)}}{N}\right]-\frac{A_t^{(N)}}{N})(\overline{M}+\overline{R})}{\sqrt{\log\log N}}\\ &\geq\liminf_{N\to\infty}\frac{\sqrt{N}(1-\rho_N)}{\sqrt{\log\log N}}>\frac{\sqrt{2}(\sigma_m+\sigma_r)}{\sqrt{\overline{M}+\overline{R}}}\text{ w.p.1.}\end{aligned}\tag{20}$$

For both cases, combined with Eqs. (16) and (17), we can get that

$$\limsup_{N\to\infty}\frac{\sum_{j=1}^{A_{t-1}^{(N)}}R_j^{(t-1)}+\sum_{j=1}^{A_t^{(N)}}M_j^{(t)}-N}{\sqrt{N\log\log N}}<0\text{ w.p.1.}\tag{21}$$

The following proof is similar to the proof of Theorem 1. ■

**Remark 3.** *If we change $\rho_N$ to $\frac{A_t^{(N)}}{N}(\overline{M}+\overline{R})$ in assumption (A7) and the assumption also holds w.p.1, then the same applies for Theorem 2 without assumption (A8).*

**Remark 4.** *Although the proof is given under MapReduce framework (2 phases), the results in Theorem 1 and Theorem 2 can be directly generalized to multiple-phases scenarios.*

**Remark 5.** *There is no assumption on the dependence between the workload of Map and Reduce for each job, thus the results in Theorem 1 and Theorem 2 can be applied in the scenario, where there could be arbitrary dependencies between the workload of Map and Reduce for each job.*

## IV. Asymptotic Optimality of Work-conserving Schedulers in Non-preemptive and Non-parallelizable Scenario

In the previous section, we show the asymptotic optimality of work-conserving schedulers in the preemptive and parallelizable scenario. In this section, we will show the performance of work-conserving scheduler in non-preemptive and non-parallelizable scenario.

**Theorem 3.** *In the non-preemptive and non-parallelizable scenario of MapReduce framework, under assumptions (A1), (A3)-(A6), and (A9), any work-conserving scheduler is asymptotically optimal.*

To prove Theorem 3, we first consider the Reduce only jobs scenario and show Lemma 2.

**Lemma 2.** *In the non-preemptive and non-parallelizable scenario, if the jobs are Reduce only and each job only has one task, then, under assumptions (A1) and (A3)-(A6), any work-conserving scheduler is asymptotically optimal.*

*Proof:* For the queue length of the one-phase jobs $Q_t^{(N)}$ at time slot $t$ and given number of machines $N$, if the queue length satisfies that

$$\limsup_{N\to\infty}\frac{Q_t^{(N)}}{N}<1\text{ w.p.1.}\tag{22}$$

for any time slot $t$, then Lemma 2 can be directly proven by the same proof process of Theorem 1. To prove this result, we use mathematical induction, which includes the following two steps:

1) For $t=1$, we prove that $\limsup_{N\to\infty}\frac{Q_1^{(N)}}{N}<1$, w.p.1.

When $t=1$, there is no remaining workload from the previous time slots. If each job which arrives in time slot 1 only has 1 unit of workload, then there is no difference between preemptive and parallelizable and non-preemptive and non-parallelizable. In this case, $\limsup_{N\to\infty}\frac{W_1^{(N)}}{N}<1$ w.p.1. holds based on Theorem 1 and the fact that $W_1^{(N)}$ is equal to the queue length in this case. If not all the jobs have 1 unit of workload, then the total queue length of jobs

$Q_1^{(N)}$ is less than the total workload of all the jobs. Thus, $\limsup_{N\to\infty} \frac{Q_1^{(N)}}{N} \leq \limsup_{N\to\infty} \frac{W_1^{(N)}}{N} < 1$ w.p.1.

2) Assume that for all $s = 1, 2, ...t-1$, $\limsup_{N\to\infty} \frac{Q_s^{(N)}}{N} < 1$ w.p.1. Under this assumption, prove that for time slot $t$, $\limsup_{N\to\infty} \frac{Q_t^{(N)}}{N} < 1$ w.p.1 is also true.

Let's consider time slot $t$. $Q_t^{(N)}$ contains all the remaining jobs which are not finished from all the previous $t-1$ time slots, i.e.

$$Q_t^{(N)} = \sum_{s=1}^{t} \text{Number of jobs which arrive at time slot } s \text{ and are remained at time slot } t. \quad (23)$$

By the induction hypothesis, for all the previous $t-1$ time slots before time slot $t$, there is not delay caused because there are not enough machines to accommodate the load. Thus, the only reason that makes some jobs are not finished yet is that these jobs do not have long enough time to finish all of their workload. Thus,

$$\frac{Q_t^{(N)}}{N} = \frac{1}{N}\sum_{s=1}^{t} A^{(N)}(s, t-s+1), \quad (24)$$

where $A^{(N)}(t,w)$ is the number of jobs, which arrives at time slot $t$ and have at least $w$ units of workload.

When $N$ goes to infinity, it can be simplified as follows:

$$\begin{aligned}
&\lim_{N\to\infty} \frac{Q_t^{(N)}}{N} = \sum_{s=1}^{t} \left( \lim_{N\to\infty} \frac{A_s^{(N)}}{N} \lim_{A_s^{(N)}\to\infty} \frac{A^{(N)}(s, t-s+1)}{A_s^{(N)}} \right) \\
&= \sum_{s=1}^{t} \left( \lim_{N\to\infty} \frac{A_s^{(N)}}{N} P(R_i \geq t-s+1) \right) \text{ w.p.1} \\
&\leq \lim_{N\to\infty} \frac{A_t^{(N)}}{N} \sum_{s=1}^{\infty} P(R_i \geq s) \leq \lim_{N\to\infty} \frac{A_t^{(N)}\overline{R}}{N} \leq \rho < 1 \text{ w.p.1.}
\end{aligned} \quad (25)$$

By mathematical induction, we can get that for any given time slot $t$, the queue length $Q_t^{(N)}$ satisfies $\limsup_{N\to\infty} \frac{Q_t^{(N)}}{N} < 1$ w.p.1. Thus, Lemma 2 can be directly achieved by a similar proving process as Theorem 1, ■

If the jobs are not Reduce-only applications, then there are both Map and Reduce phases. Based on Lemma 2, we can prove Theorem 3 as follows.

*Proof of Theorem 3:* To prove the theorem, we first introduce another type of scheduler called **Scheduler with Proper Threshold (SPT)** if it schedules the Map and Reduce tasks as follows: All the Map tasks are scheduled within a pool of $N_m = N - \sum_{k=0}^{K} N_r(k)$ machines, where $K$ is the maximum number of tasks in Reduce jobs. Since each task are also *i.i.d.*, let $\overline{R_k}$ be the expectation of $k^{th}$ task, where $1 \leq k \leq K$. For the Reduce task $k$ of each job, it should be scheduled within a pool of $N_r(k)$ machines. The $K$ pools of $\{N_r(1), \cdots N_r(K)\}$ machines, which are reserved for Reduce tasks, and the pool of $N_m$, which are reserved for Map tasks, does not have intersection. Within each pool of machines, the corresponding tasks are scheduled by any work-conserving method. We choose the threshold $N_r(k)$ as $\lceil \frac{\overline{R_k}}{\overline{M}+\overline{R}} N \rceil$.

Given a total of $N$ machines in the data center, let $Q_{0,t}^{(N)}$ be the queue length of the Map jobs at time slot $t$, and $Q_{k,t}^{(N)}$ be the queue length of the $k^{th}$ Reduce task of each job. Then, if the scheduler only schedules the Map tasks of new arrving jobs and Reduce tasks which arrived in the previous time slot, we can get $\frac{Q_{0,t}^{(N)}}{N_m} < 1$ w.p.1 by the proof of Theorem 1. Also, based on Lemma 2, we can get $\frac{Q_{k,t}^{(N)}}{N_r(k)} < 1$ w.p.1 for all $1 \leq k \leq K$. Thus, following the same proof of Theorem 1, we can directly get that any scheduler in SPT is asymptotically optimal.

In fact, the threshold between Map and Reduce tasks are unnecessary when $N$ goes to infinity. Since each pool of machines can guarantee enough space for corresponding jobs when $N$ goes to infinity, then without these thresholds, there is still enough space for all the jobs. In other words, for any work-conserving scheduler in non-preemptive and non-parallelizable scenario, the queue length $Q_t^{(N)} = \sum_{k=0}^{K} Q_{k,t}^{(N)}$, then

$$\begin{aligned}
&\lim_{N\to\infty} \frac{Q_t^{(N)}}{N} = \lim_{N\to\infty} \frac{\sum_{k=0}^{K} Q_{k,t}^{(N)}}{N_m + \sum_{k=1}^{K} N_r(k,t)} \\
&\leq \max_{k=1,\ldots K} \left\{ \lim_{N\to\infty} \frac{Q_{0,t}^{(N)}}{N_m}, \lim_{N\to\infty} \frac{Q_{k,t}^{(N)}}{N_r(k,t)} \right\} < 1 \text{ w.p.1.}
\end{aligned} \quad (26)$$

Thus, by the same proof method of Theorem 1, we can directly get that any work-conserving scheduler is asymptotically optimal. More details of the proof are in our online technical report [11]. ■

**Theorem 4.** *In the non-preemptive and non-parallelizable scenario of MapReduce framework, under assumptions (A2)-(A5), and (A8)-(A10), any work-conserving scheduler is asymptotically optimal.*

*Proof.* In the Reduce only scenario, assume that each job only has one non-preemptive and non-parallelizable Reduce task. What we need to prove now is that

$$\limsup_{N\to\infty} \frac{Q_t^{(N)} - N}{\sqrt{N \log\log N}} < 0 \text{ w.p.1.} \quad (27)$$

where $Q_t^{(N)}$ is the queue length of Reduce jobs in time slot $t$ when the system has $N$ machines. If Eq. (27) holds for all $t$, Theorem 4 can be achieved by the same proof of Theorem 3.

Similar to the proof of Lemma 2, we use mathematical induction to prove Eq. (27). With $N$ machines, let $A^{(N)}(t,w)$ be the number Reduce jobs, which arrive at time slot $t$ and have at least $w$ units of workload.

First, when $t=1$, $Q_1^{(N)}=A^{(N)}(1,1)$, then we can get that

$$\begin{aligned}&\limsup_{N\to\infty}\frac{Q_1^{(N)}-N}{\sqrt{N\log\log N}}=\limsup_{N\to\infty}\frac{A^{(N)}(1,1)-N}{\sqrt{N\log\log N}}\\&=\limsup_{N\to\infty}\frac{P(R_j^{(t)}\geq 1)(\frac{A_t^{(N)}}{N}-E[\frac{A_t^{(N)}}{N}])\sqrt{N}}{\sqrt{\log\log N}}+\\&\limsup_{N\to\infty}\frac{\sqrt{N}}{\sqrt{\log\log N}}\left(\frac{P(R_j^{(t)}\geq 1)\rho_N}{\overline{R}}-1\right)\end{aligned}\tag{28}$$

By Markov's inequality, $P(R_j^{(t)}\geq 1)\leq\overline{R}$. Thus, by assumptions (A8) and (A10), we can directly get that Eq. (27) holds for $t=1$.

Assume that Eq. (27) holds for all $s=1,2,...t-1$. If we can prove that Eq. (27) also holds for $t$, then by mathematical induction, we can show that Eq. (27) holds for any given $t$. Since Eq. (27) holds for all $s=1,2,...t-1$, then no Reduce jobs are delayed because there are not enough machines to accommodate the workload. In other words, if a job arriving at time slot $s$ is still running in time slot $t$, then the workload of that job is greater than $t-s$ units.

For simplicity, we introduce $Y_j^{(t)}(w)$ as follows:

$$Y_j^{(t)}(w)\triangleq\mathbf{1}(R_j^{(t)}\geq w)=\begin{cases}1, & R_j^{(t)}\geq w\\ 0, & R_j^{(t)}<w\end{cases},\tag{29}$$

where $\mathbf{1}(\cdot)$ is a indicator function.

Then,

$$\begin{aligned}&\limsup_{N\to\infty}\frac{Q_t^{(N)}-N}{\sqrt{N\log\log N}}=\limsup_{N\to\infty}\frac{\sum_{s=1}^{t}A^{(N)}(s,t-s+1)-N}{\sqrt{N\log\log N}}\\&=\limsup_{N\to\infty}\frac{\sum_{s=1}^{t}\sum_{j=1}^{A_s^{(N)}}Y_j^{(t)}(t-s+1)-N}{\sqrt{N\log\log N}}\\&=-\liminf_{N\to\infty}\frac{(1-\rho_N)\sqrt{N}}{\sqrt{\log\log N}}+\limsup_{N\to\infty}\frac{\sqrt{A_s^{(N)}\log\log A_s^{(N)}}}{\sqrt{N\log\log N}}\bullet\\&\sum_{s=1}^{t}\limsup_{A_s^{(N)}\to\infty}\frac{\sum_{j=1}^{A_s^{(N)}}Y_j^{(t)}(t-s+1)-E[Y_j^{(t)}(t-s+1)]A_s^{(N)}}{\sqrt{A_s^{(N)}\log\log A_s^{(N)}}}\\&+\limsup_{N\to\infty}\frac{\left(\sum_{s=1}^{t}E[Y_j^{(t)}(t-s+1)]\frac{A_s^{(N)}}{N}-\overline{R}E[\frac{A_s^{(N)}}{N}]\right)\sqrt{N}}{\sqrt{\log\log N}}.\end{aligned}\tag{30}$$

For the expectation $E\left[Y_j^{(t)}(w)\right]$ and the standard derivation $\sigma_{Y_j^{(t)}(w)}$ of $Y_j^{(t)}(w)$, we have

$$E\left[Y_j^{(t)}(w)\right]=E\left[\mathbf{1}(R_j^{(t)}\geq w)\right]=P(R_j^{(t)}\geq w)\tag{31}$$

and

$$\sigma_{Y_j^{(t)}(w)}\leq\sqrt{E\left[\left(\mathbf{1}(R_j^{(t)}\geq w)\right)^2\right]}\leq 1.\tag{32}$$

By LIL and assumption (A5), we get

$$\begin{aligned}&\limsup_{N\to\infty}\frac{Q_t^{(N)}-N}{\sqrt{N\log\log N}}\leq\sqrt{2c}t-\liminf_{N\to\infty}\frac{(1-\rho_N)\sqrt{N}}{\sqrt{\log\log N}}\\&+\limsup_{N\to\infty}\frac{\left(\sum_{s=1}^{t}E[Y_j^{(t)}(t-s+1)]\frac{A_s^{(N)}}{N}-\overline{R}E[\frac{A_s^{(N)}}{N}]\right)\sqrt{N}}{\sqrt{\log\log N}}\end{aligned}\tag{33}$$

where $c$ is constant. Since $\sum_{s=1}^{t}E[Y_j^{(t)}(t-s+1)]=\sum_{s=1}^{t}P(R_j^{(t)}\geq(t-s+1))\leq\overline{R}$, by assumption (A8) we can get that

$$\limsup_{N\to\infty}\frac{Q_t^{(N)}-N}{\sqrt{N\log\log N}}\leq\sqrt{2c}t-\liminf_{N\to\infty}\frac{(1-\rho_N)\sqrt{N}}{\sqrt{\log\log N}}\tag{34}$$

Thus, by assumption (A10), we can directly get that Eq. (27) holds for $t$. By mathematical induction, for any given $t$, Eq. 27 holds. Then, the remaining proof is same as the proof of Theorem 3. □

**Remark 6.** *If we change $\rho_N$ to $\frac{A_t^{(N)}}{N}(\overline{M}+\overline{R})$ in assumption (A10) and the assumption also holds w.p.1, then the same applies for Theorem 4 without assumption (A8).*

## V. The Relationship between $T$ and $N$

In the previous sections, we have studied the behavior of asymptotic optimality of work-conserving schedulers when $N$ goes to infinity. However, in long running system, the total number of time slots can also be very large. To show that the asymptotic optimality holds when both $T$ and $N$ are large, we need to consider the relationship between $T$ and $N$, which is shown in this section.

Let $T(N)$ be a function of $N$. We show the sufficient conditions to guarantee asymptotic optimality of work-conserving schedulers in the following theorem.

**Theorem 5.** *In the preemptive and parallelizable scenario, assume (A1), (A3)-(A6) hold. Additionally, if there exists a constant $\alpha>0$, such that*

$$\lim_{N\to\infty}T(N)\exp\left(-kN\right)N^{1+\alpha}\leq 1,\tag{35}$$

*where $k=cI_W(\frac{\overline{W}}{\rho})$. $W$ is sum of two independent random variables $M$ and $R$, which have the same distribution as $\{M_i\}$ and $\{R_i\}$, respectively. $\overline{W}=\overline{M}+\overline{R}$ and $I_W(\cdot)$ are expectation and rate function of workload $W$, respectively. Assume that the moment generating function of $W$ has finite value in some neighborhood of 0. Then*

$$\lim_{N\to\infty}\frac{F^{wc}(N,T(N))}{F^{*}(N,T(N))}=1\ \textit{w.p.1.}\tag{36}$$

*Proof:* Let

$$E_N=\left\{\begin{array}{l}\text{For } t=1,2,...T(N), \text{ there exists a } t_0,\\ \text{such that } \sum_{i=1}^{A_{t_0}^{(N)}}M_i^{(t)}+\sum_{i=1}^{A_{t_0-1}^{(N)}}R_i^{(t)}>N\end{array}\right\}.\tag{37}$$

Since

$$E_n = \bigcup_{t=1}^{T(N)} \left\{ \sum_{i=1}^{A_{t_0}^{(N)}} M_i^{(t)} + \sum_{i=1}^{A_{t_0-1}^{(N)}} R_i^{(t)} > N \right\}, \quad (38)$$

by *i.i.d.* assumption, we get

$$\begin{aligned} P(E_n) &\leq \sum_{t=1}^{T(N)} P\left( \sum_{i=1}^{A_{t_0}^{(N)}} M_i^{(t)} + \sum_{i=1}^{A_{t_0-1}^{(N)}} R_i^{(t)} > N \right) \\ &= \sum_{t=1}^{T(N)} P\left( \sum_{i=1}^{A_t^{(N)}} W_i^{(t)} > N \right), \end{aligned} \quad (39)$$

where $\{W_i^{(t)}\}$ is a sequence of i.i.d random variables that have the same distribution as $W$.

By Chernoff's inequality, we have

$$\begin{aligned} P(E_N) &\leq \sum_{t=1}^{T(N)} \exp\left( -A_t^{(N)} I_W\left(\frac{\overline{W}}{\rho}\right)\right) \\ &= T(N) \exp\left(-k_t^{(N)} N\right), \end{aligned} \quad (40)$$

where $I_W(\cdot)$ is the rate function of $W$ and $k_t^{(N)} = \frac{A_t^{(N)}}{N} I_W(\frac{\overline{W}}{\rho})$. When $N$ goes to infinity, $k_t^N$ converges to $k = cI_W(\frac{\overline{W}}{\rho})$ for all $t$. Then, when $N$ goes to infinity, we have

$$\lim_{N\to\infty} P(E_N) N^{1+\alpha} \leq \lim_{N\to\infty} T(N) \exp\left(-kN\right) N^{1+\alpha} \leq 1. \quad (41)$$

Thus, $\sum_{n=1}^{N} P(E_n) < \infty$. Based on Borel-Cantelli Lemma [10], we can get that

$$P\left( \limsup_{N\to\infty} E_N \right) = 0, \quad (42)$$

which means that $P(E_N \text{ infinitely often}) = 0$. Thus, $P(E_N \text{ only happens for finite N}) = 1$. Then, there exists a $N_0$, such that for any $N > N_0$, $P(E_N^c) = 1$.

Now let us see what is the meaning of $E_N^c$. Based on Eq. (38), we have $E_N^c = \bigcap_{t=1}^{T(N)} \{\sum_{i=1}^{A_{t_0}^{(N)}} M_i^{(t)} + \sum_{i=1}^{A_{t_0-1}^{(N)}} R_i^{(t)} > N\}$. In other words, for any event in $E_N^c$, there are always enough machines to accommodate the workload in each time slot. Thus, $E_N^c \subseteq \{\frac{F^{wc}(N,T)}{F^*(N,T)} = 1\}$. Since $P(E_N^c, \text{where } N > N_0) = 1$, then $P(\frac{F^{wc}(N,T)}{F^*(N,T)} = 1, \text{where } N > N_0) = 1$. Then we can have that

$$\lim_{N\to\infty} \frac{F^{wc}(N, T(N))}{F^*(N, T(N))} = 1 \text{ w.p.1.} \quad (43)$$

■

**Remark 7.** *For a constant $T$, we can easily check that Eq. (35) is satisfied.*

## VI. Simulation Results

### A. Simulation Setting

We evaluate the efficacy of different schedulers for both preemptive and parallelizable and non-preemptive and non-parallelizable scenarios for different number of machines $N$. We choose Poisson process as the job arrival process [5] [7]. To show the performance of work-conserving schedulers under *heavy tailed distributions*, we choose Pareto distribution as the workload distributions of Map and Reduce [12]. The cumulative distribution function of a random variable $X$ with Pareto distribution is shown as follows:

$$P(X > x) = \begin{cases} 1 - (\frac{x_m}{x})^\alpha & x \geq x_m \\ 0 & x < x_m \end{cases}, \quad (44)$$

where $x_m$ is a scale parameter and $\alpha$ is a shape parameter.

For the workload distribution of Map phase of jobs, we choose the scale parameter $x_m$ as 20, and the shape parameter $\alpha$ as 3. For the workload distribution of Reduce phase of jobs, we choose the scale parameter $x_m$ as 10, and the shape parameter $\alpha$ as 3.

We compare 4 typical work-conserving schedulers:

The **FIFO** scheduler: It is the default scheduler in Hadoop. All the jobs are scheduled in their order of arrival.

The **Fair** scheduler: It is a widely used scheduler in Hadoop. The assignment of machines are scheduled to all the waiting jobs in a fair manner. If some jobs need fewer machines, then the remaining machines are scheduled to other jobs, to avoid resource wastage and to keep the scheduler work-conserving.

The **ASRPT** scheduler: ASRPT (Available Shortest Remaining Precessing Time) is given in [7], which is based on SRPT [13] and provides good performance guarantee for any given number of machines $N$.

The **LRPT** scheduler: Jobs with larger unfinished workload are always scheduled first. Roughly speaking, the performance of this scheduler represents in a sense how poorly even some work-conserving schedulers can perform.

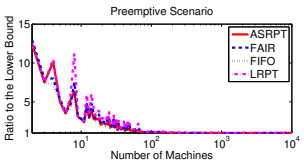
In the simulations, the ratio of a scheduler is obtained by the total flow-time of the scheduler over the lower bound of the total flow-time in 100 time slots. Here, we choose the total flow time when all the jobs can be immediately served as the lower bound. The lower bound is just the total flow time of jobs when there is infinity number of machines. Obviously, there are better (larger) lower bound estimations [7]. However, when $N$ is large enough, as we will see, the performance of all work-conserving schedulers converge to this lower bound of the flow time, showing asymptotic optimality.

### B. Fixed Traffic Intensity

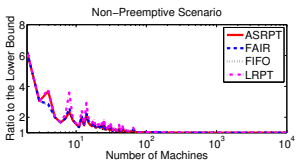
In this part, we choose traffic intensity $\rho$ as 0.5. The ratios to the lower bound of different schedulers are shown in Fig. 1 for independent Map and Reduce workload for each job. Also, we choose the correlation coefficient is 1 and $-1$ between the workload of Map and Reduce for each job as positive and negative correlation scenarios. The ratios are shown in Fig. 2 and Fig. 3, respectively. From Fig. 1 to Fig. 3, we can directly get that the ratio is very close to 1 in all the figures, when $N$ is large (e.g., $N = 500$, the difference from the lower bound is less than $0.3\%$). Thus, all the four work-conserving schedulers are asymptotically optimal in the fixed traffic intensity scenario.

### C. Heavy Traffic Scenario

In the heavy traffic scenario, we retain all the parameters as with the fixed traffic intensity scenario, except that the

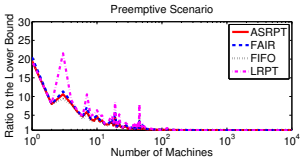



(a) Preemptive

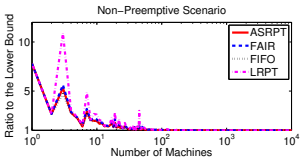



(b) Non-Preemptive

Fig. 1. Ratio to the Lower Bound (Pareto Distribution, Fixed Traffic Intensity, Map and Reduce are Independent)




(a) Preemptive



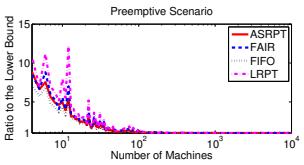

(b) Non-Preemptive

Fig. 2. Ratio to the Lower Bound (Pareto Distribution, Fixed Traffic Intensity, Map and Reduce are Positively Correlated)

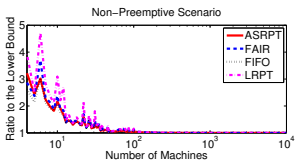
traffic intensity changes corresponding to different number of machines. We choose $\frac{(1-\rho_N)N^{\frac{1}{3}}}{\sqrt{\log\log N}} = 1$, which satisfies assumptions (A7) and (A10). Then, the ratios to the lower bound of different schedulers are shown in Fig. 4 (independent), Fig. 5 (positively correlated), and Fig. 6 (negatively correlated). Similarly, from Fig. 4 to Fig. 6, we can see that all the four work-conserving schedulers are asymptotically optimal in the heavy traffic scenario (the traffic intensity is about 0.93 when total number of machines is 10000). We find that for a wide range of different parameters and distributions, the performance is similar for both fixed traffic intensity and heavy traffic scenarios, as can be seen in our technical report [11].

## VII. Conclusion

This work is motivated by growth in the size and number of data centers. We prove that any work-conserving scheduler is asymptotically optimal under a wide range of traffic loads, including the heavy traffic limit. These results are shown for scenarios where data migration is costly (non-preemptive and non-parallelizable) and cheap (preemptive and parallelizable). We also verify these analytical results via extensive simulations, whereby state-of-the-art work-conserving schedulers are shown to have similar and close-to-optimal delay performance when the number of machines is large. Our results suggest that for large data centers, there is little to be gained by designing schedulers that optimize beyond ensuring the work conserving principle. Hence, this work provides a clear guideline on developing simple and scalable schedulers for large data centers.

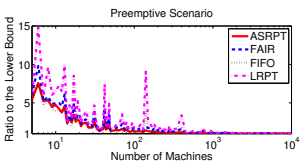



(a) Preemptive

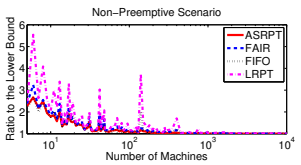



(b) Non-Preemptive

Fig. 3. Ratio to the Lower Bound (Pareto Distribution, Fixed Traffic Intensity, Map and Reduce are Negatively Correlated)

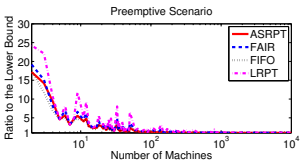



(a) Preemptive

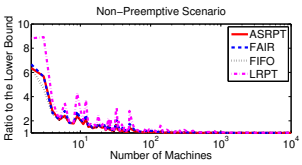



(b) Non-Preemptive

Fig. 4. Ratio to the Lower Bound (Pareto Distribution, Heavy Traffic, Map and Reduce are Independent)

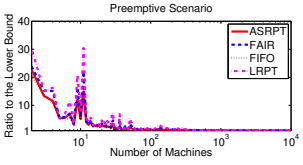



(a) Preemptive

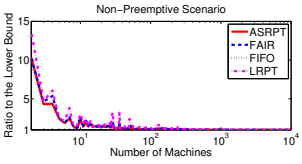



(b) Non-Preemptive

Fig. 5. Ratio to the Lower Bound (Pareto Distribution, Heavy Traffic, Map and Reduce are Positively Correlated)




(a) Preemptive




(b) Non-Preemptive

Fig. 6. Ratio to the Lower Bound (Pareto Distribution, Heavy Traffic, Map and Reduce are Negatively Correlated)

## References


[1] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," in *Proceedings of Sixth Symposium on Operating System Design and Implementation, OSDI*, pp. 137–150, December 2004.

[2] F. Chen, M. Kodialam, and T. Lakshman, "Joint scheduling of processing and shuffle phases in mapreduce systems," in *Proceedings of IEEE Infocom*, March 2012.

[3] H. Chang, M. Kodialam, R. R. Kompella, T. V. Lakshman, M. Lee, and S. Mukherjee, "Scheduling in mapreduce-like systems for fast completion time," in *Proceedings of IEEE Infocom*, pp. 3074–3082, March 2011.

[4] M. Zaharia, D. Borthakur, J. S. Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, "Job sechduling for multi-user mapreduce clusters," tech. rep., University of Califonia, Berkley, April 2009.

[5] J. Tan, X. Meng, and L. Zhang, "Performance analysis of coupling scheduler for mapreduce/hadoop," in *Proceedings of IEEE Infocom*, March 2012.

[6] B. Moseley, A. Dasgupta, R. Kumar, and T. Sarlos, "On scheduling in map-reduce and flow-shops," in *Proceedings of the 23rd ACM symposium on Parallelism in algorithms and architectures, SPAA*, pp. 289–298, June 2011.

[7] Y. Zheng, N. Shroff, and P. Sinha, "A new analytical technique for designing provably efficient mapreduce schedulers," in *Proceedings of IEEE Infocom*, pp. 1648–1656, March 2013.

[8] Y. Zheng, N. Shroff, and P. Sinha, "Performance analysis of work-conserving schedulers in minimizing the total flow-time with phase precedence," in *Proceedings of 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton'12)*, October 2012.

[9] W. Whitt, *Stochastic-Process Limits*. New York, NY, USA: Springer-Verlag New York, Inc., 2002.

[10] P. Billingsley, *Probability and Measure*. New York, NY, USA: John Wiley & Sons, inc., 1995.

[11] Y. Zheng, N. Shroff, R. Srikant, and P. Sinha, "Asymptotical Optimality of MapReduce Scheduler in Data Centers," tech. rep., June 2014.

[12] J. Tan, Y. Wang, W. Yu, and L. Zhang, "Non-work-conserving effects in mapreduce: Diffusion limit and criticality," *SIGMETRICS Perform. Eval. Rev.*, vol. 42, pp. 181–192, June 2014.

[13] K. R. Baker and D. Trietsch, *Principles of Sequencing and Scheduling*. Hoboken, NJ, USA: John Wiley & Sons, 2009.