Speed Scaling with Multiple Servers under a Sum-Power Constraint

Rahul Vaze*
School of Technology and Computer Science
Tata Institute of Fundamental Research
rahul.vaze@gmail.com

Jayakrishnan Nair
Department of Electrical Engineering
IIT Bombay
jayakrishnan.nair@iitb.ac.in

ABSTRACT

The problem of scheduling jobs and choosing their respective speeds with multiple servers under a sum-power constraint to minimize the flow time + energy is considered. This problem is a generalization of the flow time minimization problem with multiple unit-speed servers, when jobs can be parallelized, however, with a sub-linear, concave speedup function $k^{1/\alpha}, \alpha > 1$ when allocated k servers, i.e., jobs experience diminishing returns from being allocated additional servers. When all jobs are available at time 0, we show that a very simple algorithm EQUI, that processes all available jobs at the same speed is $\left(2 - \frac{1}{\alpha}\right) \frac{2}{\left(1-\left(\frac{1}{\alpha}\right)\right)}$-competitive, while in the general case, when jobs arrive over time, an LCFS based algorithm is shown to have a constant (dependent only on α) competitive ratio.

1. INTRODUCTION

Scheduling jobs with multiple servers to minimize the sum of their response times (called the flow time) is an important practical problem, and finding optimal algorithms remains challenging. An added feature in modern servers is their ability to work at different speeds. This paradigm is called *speed scaling* [2, 6, 17, 18], where one or more servers with tuneable speed are available, and operating any server at speed s consumes energy at rate $P(s)$, a non-decreasing convex function of s. With speed scaling, the problem is to choose speed of operation so as to minimize the sum of the flow time and energy.

In prior work, speed scaling problem with multiple servers has been considered [2, 6, 17, 18], however, with a fixed number of servers and without any upper bound on the power consumption of any server. For a single server, the flow time + energy problem under a power constraint or upper limit on speed has been solved in [2]. In this paper, we consider the speed scaling problem to minimize the sum of the flow time plus energy with infinite servers under a sum-power constraint across all servers. We refer to this as the *flow time + energy* problem. Even though there are unlimited number of servers, each job can only be processed by one server at any time. A special case of this problem is to minimize just the flow time, called the *flow time* problem.

The flow time problem is also equivalent to the problem of scheduling parallelizable jobs [10] with sub-linear speedup (called SUB-LINEAR-SCHED problem) described as follows. Let there be N servers with unit speed, and jobs arriving over time with different sizes have to be assigned a set of servers, so as to minimize the flow time. Jobs receive a concave, sub-linear speedup from parallelization: decreasing marginal benefit from being allocated additional servers. In particular, if $k \leq N$ is the number of servers assigned to a job, then the resulting speed obtained is $k^{1/\alpha}$ for $\alpha > 1$. When jobs can be completely parallelizable $\alpha = 1$, processing the job with shortest remaining processing time (SRPT) on all servers is known to be optimal.

In this paper, we consider online algorithms (that have only causal job arrival information) for solving the flow time + energy problem. To quantify the performance of an online algorithm, we consider the metric of competitive ratio, that is defined as the ratio of the flow time of the online algorithm and the optimal offline algorithm OPT maximized over all possible inputs (worst case).

Prior Work The SUB-LINEAR-SCHED problem has been an object of immense interest [3, 4, 11, 7, 8, 1], where practical algorithms include packing based [19], and resource reservation algorithms [14]. Heuristic policies with only numerical performance analysis can be found in [13]. In past, this problem has been considered for the combinatorial discrete allocation model [11], where an integer number of servers are assigned to any job, as well as the continuous allocation model [7, 8, 1, 3, 4], that treats the N servers as a single resource block which can be partitioned into any size and assigned to any job.

For the discrete allocation model, in [11], a variant of the SRPT algorithm is shown to be $4^{1/(1-1/\alpha)} \log W$ competitive, where $W = w_{\max}/w_{\min}$ is the ratio of the largest and the smallest job size. For the continuous allocation model, this problem has been considered in [7, 8, 1, 3, 4], where with resource augmentation, i.e., the online algorithm is allowed more resources, e.g., faster machines, than the OPT, algorithms with constant competitive ratios have been derived as a function of the resource augmentation factor.

A special case of the problem in the continuous allocation model has been considered in [4] recently, where all jobs arrive together/are available at time 0. In this simpler setting, [4] derived an optimal algorithm, called heSRPT, that gave an explicit expression for the number of servers to be dedicated for each job, which prefers smaller jobs, but unlike SRPT, all jobs are given non-zero speed. In the stochastic setting, where jobs arrive over time that have exponentially distributed job sizes, [3] showed that an algorithm called EQUI that dedicates equal number of resources to all out-

*We acknowledge support of the Department of Atomic Energy, Government of India, under project no. RTI4001

.

standing jobs is optimal to minimize the expected flow time.

Our contributions for the flow time problem are as follows.

- We first consider the setting similar to [4], where all jobs are available at time 0, and show that the well known algorithm called EQUI, that allocates equal number of resources to all outstanding jobs has competitive ratio of $\left(2-\frac{1}{\alpha}\right)\frac{1}{\left(1-\left(\frac{1}{\alpha}\right)\right)}$. For example, for $\alpha=2$, the bound is at most 3. As α increases, speeds chosen by EQUI and the optimal algorithm (heSRPT (6) [4]) converge, and that is also reflected in the competitive ratio bound that improves as α increases. In constrast to heSRPT, the utility of EQUI is that it does not need to know the exact remaining sizes of the jobs, and the speed is identical for all jobs which makes it easy to implement in practice.

- For the online setting, where jobs arrive over time, we propose an algorithm called the FRACTIONAL-LCFS-EQUI that processes a fraction of the outstanding jobs that have arrived most recently, with equal speed. We show that FRACTIONAL-LCFS-EQUI has a competitive ratio that depends only on α and not on system parameters such as the total number of jobs, and their respective sizes. In prior work, for similar results, resource augmentation was needed [7, 8, 1]. Thus, our result overcomes a fundamental bottleneck of resource augmentation compared to [7, 8, 1], however, it must be noted that [7, 8, 1] considered more complicated problem setups as discussed earlier.

- For the more general flow time + energy problem, the competitive ratio bound is at most twice their counterparts for the flow time problem in the two respective cases.

2. PROBLEM FORMULATION

Let there be an unlimited supply of servers, however, any job can be processed on any one server at any time (no parallelization is allowed). We consider that preemption is allowed, i.e., a job can be halted at any time and restarted later on any server. Each server when executing a job at speed $s(t)$ at time t, consumes power $P(s(t))$. Let $P(x)=x^\alpha$ where $\alpha>1$. In addition, there is a sum-power constraint of $\mathbf{p}$ across all servers, i.e., $\sum_{i:s_i(t)>0} P(s_i(t)) \le \mathbf{p}$, for all t, where $s_i(t)$ is the speed of an active server i at time t.

If a server works with speed s for a time duration t on a job, then st amount of work is completed for that job. Set of jobs $\mathcal{J}$ arrive over time, where an arriving job $j\in\mathcal{J}$ with size w_j is defined to be complete at time d_j, if w_j amount of work has been completed for it by time d_j, where the work could have been done by different servers at different times. Hence the flow time is $\sum_{j\in\mathcal{J}}(d_j-a_j)=\int n(t)dt$, where $n(t)$ is the number of unfinished jobs in the system at time t.

The flow time problem is then

$$\min \int n(t)dt, \tag{1}$$

subject to $\sum_{i:s_i(t)>0} P(s_i(t)) \le \mathbf{p}$, while the flow time + energy problem is

$$\min \int n(t)dt + \int \sum_{i:s_i(t)>0} P(s_i(t))dt, \tag{2}$$

subject to $\sum_{i:s_i(t)>0} P(s_i(t)) \le \mathbf{p}$.

REMARK 1. *Note that even under the sum-power constraint, the metric of flow time + energy is meaningful, since it is not necessary that the energy used by an optimal algorithm is equal to the maximum possible allowed by the constraint. For example, when the number of outstanding jobs is small, an optimal algorithm may choose a small speed such that the total power consumed is less than the sum-power constraint.*

Next, we show that problem (1) is equivalent to the SUB-LINEAR-SCHED problem, that has N parallel and identical servers. Similar to [7, 4], we consider the continuous allocation model, where N is treated as a single resource block which can be divided into chunks of arbitrary sizes and allocated to different jobs. Any job is parallelizable with concave speedup, i.e., if job j is allotted $k_j(t)$ number of servers at time t, then the service rate experienced by job j at time t is $s_j(t)=S(k_j(t))=k_j(t)^{1/\alpha}$, where $\alpha>1$. Here, $S(\cdot)$ denotes the speedup function, that is concave. The parameter α controls the parallelizability of any job, and depending on α, jobs experience appropriate diminishing returns from being allocated additional servers. Note that

$$\sum_{j=1}^{A(t)} k_j(t) \le N, \tag{3}$$

where $A(t)$ is the set of jobs that are given non-zero service rate at time t. Note that the objective is to minimize the flow time of all jobs.

To cast the SUB-LINEAR-SCHED problem as a flow time problem (1), suppose that for each job we can 'create' its own dedicated server, and a job is processed on only one server, and cannot be parallelized. Let $s_j(t)$ denote the speed allocated to the server processing job j at time t. Let $k_j(t)=P(s_j(t)):=S^{-1}(s_j(t))$ be the power consumption of job j on its own server if it is processed at speed $s_j(t)$, where $P(s)=s^\alpha, \alpha>1$. Then, (3) is equivalent to $\sum_{j\in A(t)} P(s_j(t)) \le N$, the total power used across all the active servers is at most N.

In both these models, the service speed of job j is s_j, so the flow times across both the models would be identical. Letting $N=\mathbf{p}$, we see that SUB-LINEAR-SCHED is equivalent to the flow time problem (1).

In the following, we will consider Problem (2), and propose algorithms and bound their competitive ratios (4). Minimal changes to be made for algorithms to be feasible, and analysis to be applicable for Problem (1) are mentioned in Remark 3. Consequently, we will only indicate the corresponding competitive ratio results for Problem (1).

Metric We represent the optimal offline algorithm (that knows the entire job arrival sequence in advance) as OPT. Let $n(t)$ $(n_o(t))$ and $P_{\text{sum}}(t)$ $(P^o_{\text{sum}}(t))$ be the number of outstanding jobs with an online algorithm $\mathcal{A}$ (OPT), and the sum of the power used by an online algorithm $\mathcal{A}$ (OPT) across all servers at time t, respectively. For Problem (2), we will consider the metric of competitive ratio which for an algorithm $\mathcal{A}$ is defined as

$$\mu_{\mathcal{A}} = \max_{\sigma} \frac{\int (n(t)+P_{\text{sum}}(t))dt}{\int (n_o(t)+P^o_{\text{sum}}(t))dt}, \tag{4}$$

where σ is the input sequence consisting of jobs set $\mathcal{J}$. Since σ is arbitrary, we are not making any assumption on the job arrival times, or their sizes.

We will propose an online algorithm $\mathcal{A}$, and bound $\mu_{\mathcal{A}} \leq \kappa$, by showing that for each time instant t

$$n(t) + P_{\text{sum}}(t) + d\Phi(t)/dt \leq \kappa(n_o(t) + P^o_{\text{sum}}(t)), \quad (5)$$

where $\Phi(t)$ is some function called the **potential function** that satisfies the boundary conditions:

- $\Phi(t) = 0$ initially before all job arrivals and $\Phi(\infty) = 0$.
- $\Phi(t)$ does not increase on any job arrival or job departure with the algorithm or the OPT.

Integrating (5) over time, implies that the competitive ratio of $\mathcal{A}$ is at most κ.

3. ALL JOBS AVAILABLE AT TIME 0

We first consider the simpler setting when all jobs of $\mathcal{J}$ arrive together at time 0. For Problem (1), this setting has been considered recently [4], and an optimal algorithm (called heSRPT) has been derived. In particular, with heSRPT, let at time t there are $n(t)$ unfinished jobs that are indexed in decreasing order of their remaining sizes, $w_{n(t)}(t) \leq \cdots \leq w_2(t) \leq w_1(t)$. Then the number of servers dedicated to job $i = 1, \ldots, n(t)$ is

$$k_i(t) = N\left(\left(\frac{i}{n(t)}\right)^{\left(\frac{1}{1-1/\alpha}\right)} - \left(\frac{i-1}{n(t)}\right)^{\left(\frac{1}{1-1/\alpha}\right)}\right), \quad (6)$$

and corresponding speed is $S(k_i(t)) = k_i(t)^{1/\alpha}$. Thus, shorter jobs get more servers, and consequently more speed.

We show that a simpler algorithm, called EQUI, that processes all jobs simultaneously at the same speed, and does not require the knowledge of remaining job sizes, has a constant (depending on α) competitive ratio for both Problem (1) and Problem (2).

We begin with some preliminaries. Let $Q(x) = \frac{x}{P^{-1}(x)}$. For $P(x) = x^\alpha$, $Q(x) = x^{1-\frac{1}{\alpha}}$.

Lemma 2. *With $P^o_{sum}(t)$ as the sum-power used by* OPT *at any time t, the maximum speed devoted to processing any one job by the* OPT *is at most $P^{-1}(P^o_{sum}(t))$. Moreover, the sum of the speeds with which* OPT *is processing any of its k jobs is at most $Q(k)P^{-1}(P^o_{sum}(t))$.*

Proof is trivial and hence omitted.

3.1 Algorithm EQUI

At time t, if the outstanding number of jobs in the system is $n(t)$, then all $n(t)$ jobs are processed parallely on $n(t)$ servers, each with identical speed

$$s(t) = P^{-1}\left(\frac{\min\{n(t), \mathbf{p}\}}{n(t)}\right). \quad (7)$$

Thus, the total power used by EQUI at any time is $\leq n(t)P\left(P^{-1}\left(\frac{\min\{n(t),\mathbf{p}\}}{n(t)}\right)\right) \leq \min\{n(t), \mathbf{p}\} \leq \mathbf{p}$.

Remark 3. *All algorithms presented in the paper when applied to Problem* (1) *will have the term* $\min\{n(t), \mathbf{p}\}$ *in their speed choice replaced by* $\mathbf{p}$. *Similarly, for potential functions defined in* (9) *and* (13), *terms* $\min\{., \mathbf{p}\}$ *will be replaced by* $\mathbf{p}$.

3.2 Potential Function

At time t, let $A(t)$ be the set of unfinished jobs with EQUI with $n(t) = |A(t)|$, and for the i^{th} job, $i \in A(t)$, let $q_i(t)$ be its remaining size. Then

$$n^i(t,q) = \begin{cases} 1 & \text{for } q \leq q_i(t), \\ 0 & \text{otherwise.} \end{cases} \quad (8)$$

Similarly, let $n_o(t)$ be the number of unfinished jobs with the OPT, and the corresponding quantity to $n^i(t,q)$ for the i^{th} job with the OPT, be denoted by $n^i_o(t,q)$.

Consider the potential function $\Phi_{sf}(t) =$

$$c_1 P^{-1}\left(\frac{n(t)}{\min\{n(t), \mathbf{p}\}}\right)\left(\sum_{i\in A(t)}\int_0^\infty (n^i(t,q) - n^i_o(t,q))^+ dq\right), \quad (9)$$

where c_1 is a constant to be chosen later, and $(x)^+ = \max\{0, x\}$.

Clearly, $\Phi_{sf}(t)$ satisfies the first boundary condition. Since all jobs are available at time 0, to check whether $\Phi_{sf}(t)$ satisfies the second boundary condition, we only need to check whether $\Phi_{sf}(t)$ increases on a departure of a job with either the EQUI or the OPT.

Lemma 4. *Potential function $\Phi_{sf}(t)$* (9) *does not increase on a departure of a job with either the EQUI or the* OPT.

Proof: On a departure of a job with the algorithm or the OPT, $n^i(t,q)$ or $n^i_o(t,q)$ changes for only $q = 0$, and since there is an integral outside, $\int_0^\infty (n^i(t,q) - n^i_o(t,q))^+ dq$ remains the same on a departure of a job with either the algorithm or the OPT.

The pre-factor term $P^{-1}\left(\frac{n(t)}{\min\{n(t),\mathbf{p}\}}\right)$ changes though, when there is a departure of a job with the algorithm, on account of $n(t) \to n(t) - 1$. Consider time t^-, just before a departure at time t, where $n(t) = n(t^-) - 1$.

Case Ia: $\min\{n(t^-), \mathbf{p}\} = \mathbf{p}$ and $\min\{n(t), \mathbf{p}\} = \mathbf{p}$. In this case, $P^{-1}\left(\frac{n(t^-)}{\min\{n(t^-),\mathbf{p}\}}\right) - P^{-1}\left(\frac{n(t)}{\min\{n(t),\mathbf{p}\}}\right) > 0$.

Case Ib: $\min\{n(t^-), \mathbf{p}\} = \mathbf{p}$ and $\min\{n(t), \mathbf{p}\} = n(t)$. In this case, $P^{-1}\left(\frac{n(t^-)}{\min\{n(t^-),\mathbf{p}\}}\right) - P^{-1}(1) \geq 0$

Case II: $\min\{n(t^-), \mathbf{p}\} = n(t^-)$ In this case,

$$P^{-1}\left(\frac{n(t^-)}{\min\{n(t^-), \mathbf{p}\}}\right) - P^{-1}\left(\frac{n(t)}{\min\{n(t), \mathbf{p}\}}\right) = 0.$$

Thus, the pre-factor $P^{-1}\left(\frac{n(t)}{\min\{n(t),\mathbf{p}\}}\right)$ does not increase at any job departure. Moreover, the departure of any job with the OPT does not change the pre-factor. Since the integral is always non-negative, the result follows. □

Next, we characterize the drift $d\Phi_{sf}(t)/dt$.

Lemma 5. $d\Phi_{sf}(t)/dt \leq -c_1((-\max\{n(t) - n_o(t), 0\})$

$$+ c_1\left(\frac{1}{\alpha}\right)\max\{n(t), P^o_{sum}(t)\} + c_1\left(1 - \frac{1}{\alpha}\right)n_o(t).$$

All missing proofs can be found in the longer version of the paper [16].

Theorem 6. *The competitive ratio of EQUI for Problem* (2) *when all jobs are available at time* 0, *is at most*

$$\mu(\alpha) = \left(2 - \frac{1}{\alpha}\right)\frac{2}{(1 - (\frac{1}{\alpha}))}.$$

For $\alpha = 2$, $\mu(2) = 6$. Moreover, $\mu(\alpha)$ is a decreasing function of $\alpha > 1$.

Proof: Case I : $\max\{n(t) - n_o(t), 0\} = 0$. In this case, from Lemma 5, we can write (5), as $n(t) + P_{\text{sum}}(t) + d\Phi(t)/dt$

$$
\begin{aligned}
&= n(t) + \min\{n(t), \mathbf{p}\} + c_1\left(\frac{1}{\alpha}\right)\max\{n(t), P^o_{\text{sum}}(t)\} \\
&\quad + c_1\left(1 - \frac{1}{\alpha}\right)n_o(t), \\
&\le 2n(t) + c_1\left(\frac{1}{\alpha}\right)(n(t) + P^o_{\text{sum}}(t)) + c_1\left(1 - \frac{1}{\alpha}\right)n_o(t), \\
&\overset{(a)}{\le} (2 + c_1)n_o(t) + c_1\left(\frac{1}{\alpha}\right)P^o_{\text{sum}}(t), \\
&\le (2 + c_1)(n_o(t) + P^o_{\text{sum}}(t)), \qquad (10)
\end{aligned}
$$

where (a) follows since $n(t) \le n_o(t)$.

Case II: $n_o(t) > 0$, and $\max\{n(t) - n_o(t), 0\} = n(t) - n_o(t)$. Using Lemma 5, we can write (5), as $n(t) + P_{\text{sum}}(t) + d\Phi(t)/dt$

$$
\begin{aligned}
&= n(t) + \min\{n(t), \mathbf{p}\} - c_1(n(t) - n_o(t)) \\
&\quad + c_1\left(\frac{1}{\alpha}\right)\max\{n(t), P^o_{\text{sum}}(t)\} + c_1\left(1 - \frac{1}{\alpha}\right)n_o(t), \\
&\le 2n(t) - c_1(n(t) - n_o(t)) + c_1\left(\frac{1}{\alpha}\right)(n(t) + P^o_{\text{sum}}(t)) \\
&\quad + c_1\left(1 - \frac{1}{\alpha}\right)n_o(t), \\
&\le n(t)\left(2 - c_1 + c_1\left(\frac{1}{\alpha}\right)\right) + n_o(t)\left(c_1\left(1 + \left(1 - \frac{1}{\alpha}\right)\right)\right) \\
&\quad + c_1\left(\frac{1}{\alpha}\right)P^o_{\text{sum}}(t), \\
&\le \left(c_1\left(2 - \frac{1}{\alpha}\right)\right)(n_o(t) + P^o_{\text{sum}}(t)) \qquad (11)
\end{aligned}
$$

for $c_1 \ge 2/\left(1 - \left(\frac{1}{\alpha}\right)\right)$. When $n_o(t) = 0$, then we do not have to add the contribution of the OPT from Lemma 5, and we get $n(t) + P_{\text{sum}}(t) + d\Phi(t)/dt$

$$
\begin{aligned}
&= n(t) + n(t)\left(\frac{\min\{n(t), \mathbf{p}\}}{n(t)}\right) - c_1 n(t), \\
&\le n(t)(2 - c_1) \le 0, \qquad (12)
\end{aligned}
$$

for $c_1 = 2$. Thus, choosing $c_1 = 2/\left(1 - \left(\frac{1}{\alpha}\right)\right)$, from (10), (11), and (12), (5) holds for $\kappa = \left(2 - \frac{1}{\alpha}\right)\frac{2}{\left(1 - \left(\frac{1}{\alpha}\right)\right)}$. □

Identical proof follows for Problem (1), since essentially, the only difference when considering Problem (1) is that we can remove the energy term corresponding to $P_{\text{sum}}(t)$, and choose $c_1 = 1/\left(1 - \left(\frac{1}{\alpha}\right)\right)$, and show that (5) holds for $\kappa = \left(2 - \frac{1}{\alpha}\right)\frac{1}{\left(1 - \left(\frac{1}{\alpha}\right)\right)}$.

THEOREM 7. *The competitive ratio of EQUI for Problem (1) when all jobs are available at time 0, is at most $\left(2 - \frac{1}{\alpha}\right)\frac{1}{\left(1 - \left(\frac{1}{\alpha}\right)\right)}$. For $\alpha = 2$, the upper bound is at most 3, and decreases to 2 as α increases.*

Discussion: In this section, we showed that for minimizing flow time (Problem (1)), a simpler algorithm (EQUI) than the optimal heSRPT algorithm [4], that processes all available jobs with the same speed, is constant (depending only on α) competitive. Moreover, as α increases, speeds chosen by EQUI and heSRPT algorithm (6) converge, and that is also reflected in the competitive ratio bound that improves as α increases. Thus, knowing job sizes and using job dependent speed is not critical for staying close to the optimal performance. The utility of EQUI is that it does not need to know the exact remaining sizes of the jobs, thus making it applicable for more wider network setting where pipelining [12, 5, 15] is implemented, and jobs on arrival do not reveal their true sizes.

4. ONLINE JOB ARRIVALS

In this section, we consider Problem (2) in the online case, where jobs arrive over time with arbitrary sizes and at arbitrary time instants.

4.1 Algorithm FRACTIONAL-LCFS-EQUI

At time t, let the outstanding number of jobs in the system be $n(t)$. **Scheduling:** Process the $\beta n(t), \beta < 1$, jobs that have arrived **most recently** in their respective $\beta n(t)$ servers. [1] **Speed:** Use EQUI, to process all the $\beta n(t)$ jobs at equal speed $s(t) = P^{-1}\left(\frac{\min\{n(t), \mathbf{p}\}}{\beta n(t)}\right)$.

By its very definition, FRACTIONAL-LCFS-EQUI satisfies the total power constraint as follows $P_{\text{sum}}(t)$

$$\le \beta n(t) P\left(P^{-1}\left(\frac{\min\{n(t), \mathbf{p}\}}{\beta n(t)}\right)\right) \le \min\{n(t), \mathbf{p}\} \le \mathbf{p}.$$

REMARK 8. *Choosing $\beta = 1$,* FRACTIONAL-LCFS-EQUI *is identical to EQUI. Intuitively following heSRPT algorithm (6) at each time t in the online case appears better than* FRACTIONAL-LCFS-EQUI, *since it is locally optimal, however, analyzing the heSRPT algorithm in the online case appears challenging.*

The main result of this section is as follows.

THEOREM 9. *For any $\alpha > 1$, there exists a $\beta < 1$, such that the competitive ratio of algorithm* FRACTIONAL-LCFS-EQUI *for Problem (2) is a constant (depends only on α) and is independent of the number of jobs, and their sizes. For example, for $2 \le \alpha \le 3$, with $\beta = \frac{1}{6}$, the competitive ratio is at most 693.*

We get the result for Problem (1) as a corollary as follows.

COROLLARY 10. *For any $\alpha > 1$, there exists a $\beta < 1$, such that the competitive ratio of algorithm* FRACTIONAL-LCFS-EQUI *for Problem (1) is a constant (depends only on α) and is independent of the number of jobs, and their sizes. In particular, the competitive ratio will be at most half of the competitive ratio for Problem (2). For example, for $2 \le \alpha \le 3$, with $\beta = \frac{1}{6}$, the competitive ratio is at most 345.*

Discussion: Similar to EQUI, algorithm FRACTIONAL-LCFS-EQUI is also a non-clairvoyant algorithm, i.e., it does not need to know the remaining size of any outstanding job. The main novelty of Theorem 9 over previous such results [7, 8, 9], is that it is proven without needing resource augmentation. With resource augmentation, an online algorithm is given servers that are allowed to operate at speed $s(1 + \theta), \theta > 0$ while consuming only power $P(s)$, but the OPT's consumption is kept intact at $P(s)$ with speed s. Thus, an online algorithm is given extra/faster resources. In

[1] If $\beta n(t)$ is fractional, then we mean $\lceil \beta n(t) \rceil$.

[7, 8, 9], algorithms with competitive ratio as a function of α and θ have been derived for a similar but more complicated non-clairvoyant settings.

4.2 Proof of Theorem 9

From here on we refer to algorithm FRACTIONAL-LCFS-EQUI as just algorithm. Let at time t, the set of outstanding (unfinished) number of jobs with the algorithm be $A(t)$ with $n(t) = |A(t)|$. Let at time t, the **rank** $r_j(t)$ of a job $j \in A(t)$ be equal to the number of outstanding jobs of $A(t)$ with the algorithm that have arrived before job j. Note that the rank of a job does not change on arrival of a new job, but can change if a job departs that had arrived earlier.

As before, $Q(x) = \frac{x}{P^{-1}(x)}$. which specializes to $Q(x) = x^{1-\frac{1}{\alpha}}$ for $P(x) = x^{\alpha}$. Then we consider the following potential function

$$\Phi(t) = c \sum_{j \in A(t)} \frac{r_j(t)\left(w_j^A(t) - w_j^o(t)\right)^+}{P^{-1}(\min\{r_j(t), \mathbf{p}\})Q(r_j(t))}, \tag{13}$$

where $w_j^A(t)$ ($w_j^o(t)$) is the remaining size of job j with the algorithm (OPT) at time t, and c is a constant to be chosen later.

REMARK 11. *The potential function* (13) *is very similar to the one used in [9] for a very different problem. The main novelty of our result is that we avoid resource augmentation unlike [9]. Moreover, we would like to point out that for Problem* (2)*, the most popular potential functions used in [2, 17] cannot be used since they need job processing speed to be a function of $P^{-1}(n(t))$ which is not possible because of the sum-power constraint.*

We next show that the potential function $\Phi(t)$ satisfies the second boundary condition. The fact that the first boundary condition is satisfied is trivial.

LEMMA 12. *Potential function $\Phi(t)$* (13) *does not change on arrival of any new job. Moreover, on a departure of a job with the algorithm or the* OPT*, the potential function $\Phi(t)$* (13) *does not increase.*

We next bound the drift $d\Phi(t)/dt$ because of the processing by the OPT, and the algorithm, respectively. To avoid cumbersome notation, we write $\beta n(t)$ instead of $\lceil \beta n(t) \rceil$ everywhere.

LEMMA 13. *The change in the potential function* (13) *because of the* $\mathsf{OPT}'s$ *contribution*

$$d\Phi(t)/dt \le \begin{cases} cP^o_{sum}(t) & \text{if } n(t) \le \mathbf{p}, \\ cn(t)\frac{Q(n_o(t))}{Q(n(t))} & \text{if } n(t) > \mathbf{p}. \end{cases} \tag{14}$$

LEMMA 14. *For $\gamma < \beta$, when $n_o(t) \le \gamma n(t)$, the change in the potential function* (13) *because of the algorithm's contribution is*

$$d\Phi(t)/dt \le -\frac{(1-\beta)(\beta-\gamma)n(t)}{P^{-1}(\beta)}. \tag{15}$$

Proof: [Proof of Theorem 9] To prove the Theorem, we check the running condition (5) for the two cases separately : i) $n_o(t) > \gamma n(t)$ and then ii) $n_o(t) \le \gamma n(t)$, and show that it holds for a constant κ.

Case i) $n_o(t) > \gamma n(t)$. In this case, we only count the $\mathsf{OPT}'s$ contribution to $d\Phi(t)/dt$, which is sufficient since the algorithm's contribution to $d\Phi(t)/dt$ is always non-positive. When $n(t) > \mathbf{p}$, from Lemma 13, we have that (5), $n(t) + P_{\text{sum}}(t) + d\Phi(t)/dt$

$$\begin{aligned} &\le n(t) + \min\{n(t), \mathbf{p}\} + cn(t)\frac{Q(n_o(t))}{Q(n(t))}, \\ &\overset{(a)}{\le} 2n(t) + cn(t)\frac{Q(bn(t))}{Q(n(t))} \le 2n(t) + cn(t)b^{1-1/\alpha}, \\ &\overset{(b)}{\le} 2n(t) + cn(t)b = 2n(t) + cn_o(t) \overset{(c)}{\le} (2/\gamma + c)n_o(t), \end{aligned} \tag{16}$$

where in (a) we let $n_o(t) = bn(t)$. We first consider the case when $b > 1$, where inequality (b) follows when $b > 1$. Finally (c) follows since $n_o(t) > \gamma n(t)$. When $b < 1$, then $\frac{Q(bn(t))}{Q(n(t))} < 1$. Thus, similar to (16), for $b < 1$, we get $n(t) + P_{\text{sum}}(t) + d\Phi(t)/dt$

$$\begin{aligned} &\le n(t) + \min\{n(t), \mathbf{p}\} + cn(t)\frac{Q(n_o(t))}{Q(n(t))}, \\ &\le n(t)(2 + c), \\ &\le \frac{1}{\gamma}(2 + c)n_o(t). \end{aligned} \tag{17}$$

When $n(t) \le \mathbf{p}$, from Lemma 13, we have that (5), $n(t) + P_{\text{sum}}(t) + d\Phi(t)/dt$

$$\begin{aligned} &\le n(t) + \min\{n(t), \mathbf{p}\} + cP^o_{\text{sum}}(t), \\ &\le \frac{2+c}{\gamma}(n_o(t) + P^o_{\text{sum}}(t)). \end{aligned} \tag{18}$$

Combining, (17) and (18), when $n_o(t) > \gamma n(t)$

$$n(t) + P_{\text{sum}}(t) + d\Phi(t)/dt, \le \frac{1}{\gamma}(2 + c)(n_o(t) + P^o_{\text{sum}}(t)). \tag{19}$$

Case ii) $n_o(t) \le \gamma n(t)$. Let $n_o(t) > 0$. When $n(t) > \mathbf{p}$, From Lemma 13 and Lemma 14, (5) can be bounded as $n(t) + P_{\text{sum}}(t) + d\Phi(t)/dt$

$$\begin{aligned} &\le n(t) + \min\{n(t), \mathbf{p}\} \\ &\quad + cn(t)\frac{Q(n_o(t))}{Q(n(t))} - c\frac{(1-\beta)(\beta-\gamma)n(t)}{P^{-1}(\beta)}, \qquad (20) \\ &\overset{(a)}{\le} n(t)\left(2 + c\left(\gamma^{1-1/\alpha} - \frac{(1-\beta)(\beta-\gamma)}{P^{-1}(\beta)}\right)\right) \overset{(b)}{\le} 0, \qquad (21) \end{aligned}$$

where (a) follows since $n_o(t) \le \gamma n(t)$, while (b) follows for choice of γ, β, c that satisfy

$$\frac{(1-\beta)(\beta-\gamma)}{P^{-1}(\beta)} > \gamma^{1-1/\alpha} \text{ and } c \ge \frac{-2}{\left(\gamma^{1-1/\alpha} - \frac{(1-\beta)(\beta-\gamma)}{P^{-1}(\beta)}\right)}. \tag{22}$$

When $n_o(t) = 0$, the OPT's contribution is zero, and (21) holds for a smaller value of c.

Similarly, when $n(t) \le \mathbf{p}$, $n(t) + P_{\text{sum}}(t) + d\Phi(t)/dt$

$$\begin{aligned} &\le n(t) + \min\{n(t), \mathbf{p}\} + cP^o_{\text{sum}}(t) - c\frac{(1-\beta)(\beta-\gamma)n(t)}{P^{-1}(\beta)}, \\ &\le 2n(t)\left(1 - c\left(\frac{(1-\beta)(\beta-\gamma)}{P^{-1}(\beta)}\right)\right) + cP^o_{\text{sum}}(t), \\ &\overset{(a)}{\le} cP^o_{\text{sum}}(t), \end{aligned} \tag{23}$$

where (a) follows as long as $c\left(\frac{(1-\beta)(\beta-\gamma)}{P^{-1}(\beta)}\right) > 1$. Combining (19), (21) and (23), using (5), the competitive ratio of the proposed algorithm is

$$\frac{2+c}{\gamma} \tag{24}$$

where c, β and γ satisfy (22). Note that the bound (24) can be optimized by choosing the optimal value of β and γ satisfying (22). It is easy to see that depending on α, there exists a β satisfying (22) with $\gamma = \beta^2$.

For example, for $\alpha = 2, 3$, let $\beta = \frac{1}{6}$ and $\gamma = \beta^2$, and we get a competitive ratio bound of 693 and 680, respectively, as follows. In fact for $2 \le \alpha \le 3$, choosing $\beta = \frac{1}{6}$ and $\gamma = \beta^2$, the competitive ratio is at most 693. For $\alpha = 2$, let $\beta = \frac{1}{6}$, and $\gamma = \beta^2$. Then $\frac{(1-\beta)(\beta-\gamma)}{P^{-1}(\beta)} = 5/6(5/36)/\sqrt{1/6} = 2.455/6(5/36) = .283$ while $\gamma^{1-1/\alpha} = .167$. Thus, we have $\frac{(1-\beta)(\beta-\gamma)}{P^{-1}(\beta)} > \gamma^{1-1/\alpha}$ and $c = 2/(.283 - .167) = 17.24$. Thus, the competitive ratio when $\alpha = 2$ is $\frac{2+c}{\gamma} = 36 \times (2 + 17.24) < 693$. Similarly for $\alpha = 3$, with $\beta = \frac{1}{6}$, and $\gamma = \beta^2$, $c = 16.66$ and the competitive ratio upper bound is $\frac{2+c}{\gamma} = 36 \times (18.66) < 680$. □

Proof: [Proof of Corollary 10] Proof is immediate by noting that with Problem (1), we do not have to add the energy consumption term $P_{\text{sum}}(t)$ for checking the running condition (5), thus resulting in a two-fold decrease in the κ needed to satisfy (5). □

5. CONCLUSIONS

In this paper, we considered an important problem of flow time minimization in data centers, where jobs have limited parallelizability, and they experience diminishing returns from being allocated additional servers. When all jobs are available at time 0, a very simple algorithm called EQUI that processes all outstanding jobs at the same speed is shown to have a constant competitive ratio that only depends on the speed-up exponent α. For the most relevant speed-up exponents of 2 and 3, the competitive ratio is at most 3. Thus, even without knowing job-sizes, and processing all of them at the same speed, EQUI is not too sub-optimal.

For the general online setting, where jobs arrive over time, we propose a LCFS type algorithm for scheduling and EQUI for speed selection, and show that its competitive ratio is a constant that only depends on the speedup exponent α. Our result overcomes fundamental difficulty found in literature where similar results were shown only in the presence of resource augmentation, where an online algorithm is allowed more resources than the optimal offline algorithm.

6. REFERENCES

[1] K. Agrawal, J. Li, K. Lu, and B. Moseley. Scheduling parallelizable jobs online to minimize the maximum flow time. In *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 195–205, 2016.

[2] N. Bansal, H.-L. Chan, and K. Pruhs. Speed scaling with an arbitrary power function. In *Proceedings of the twentieth annual ACM-SIAM symposium on discrete algorithms*, pages 693–701. SIAM, 2009.

[3] B. Berg, J.-P. Dorsman, and M. Harchol-Balter. Towards optimality in parallel scheduling. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(2):1–30, 2017.

[4] B. Berg, R. Vesilo, and M. Harchol-Balter. heSRPT: Optimal scheduling of parallel jobs with known sizes. *SIGMETRICS Perform. Evaluation Rev.*, 47(2):18–20, 2019.

[5] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, K. Elmeleegy, and R. Sears. Mapreduce online. In *NSDI*, volume 10, page 20, 2010.

[6] N. R. Devanur and Z. Huang. Primal dual gives almost optimal energy-efficient online algorithms. *ACM Transactions on Algorithms (TALG)*, 14(1):1–30, 2017.

[7] J. Edmonds. Scheduling in the dark. *Theoretical Computer Science*, 235(1):109–141, 2000.

[8] J. Edmonds and K. Pruhs. Scalably scheduling processes with arbitrary speedup curves. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pages 685–692. SIAM, 2009.

[9] A. Gupta, S. Im, R. Krishnaswamy, B. Moseley, and K. Pruhs. Scheduling heterogeneous processors isn't as easy as you think. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete algorithms*, pages 1242–1253. SIAM, 2012.

[10] M. Harchol-Balter. Open problems in queueing theory inspired by datacenter computing. *Queueing Systems*, 97(1):3–37, 2021.

[11] S. Im, B. Moseley, K. Pruhs, and E. Torng. Competitively scheduling tasks with intermediate parallelizability. *ACM Transactions on Parallel Computing (TOPC)*, 3(1):1–19, 2016.

[12] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly. Dryad: distributed data-parallel programs from sequential building blocks. In *ACM SIGOPS operating systems review*, volume 41, pages 59–72. ACM, 2007.

[13] S.-H. Lin, M. Paolieri, C.-F. Chou, and L. Golubchik. A model-based approach to streamlining distributed training for asynchronous sgd. In *2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 306–318. IEEE, 2018.

[14] R. Ren and X. Tang. Clairvoyant dynamic bin packing for job scheduling with minimum server usage time. In *Proceedings of the 28th ACM SPAA*, pages 227–237, 2016.

[15] C. J. Rossbach, Y. Yu, J. Currey, J.-P. Martin, and D. Fetterly. Dandelion: a compiler and runtime for heterogeneous systems. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, pages 49–68. ACM, 2013.

[16] R. Vaze and J. Nair. Speed scaling with multiple servers under a sum power constraint. In *https://arxiv.org/abs/2108.06935.*

[17] R. Vaze and J. Nair. Multiple server SRPT with speed scaling is competitive. *IEEE/ACM Transactions on Networking*, 28(4):1739–1751, 2020.

[18] R. Vaze and J. Nair. Network speed scaling. *Performance Evaluation*, 144:102145, 2020.

[19] A. Verma, L. Pedrosa, M. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes. Large-scale cluster management at google with borg. In *Proceedings of the Tenth European Conference on Computer Systems*, pages 1–17, 2015.