Non-preemptive Datacenter Scheduling via Scaling Cycles

Zhongrui Chen
UNC Chapel Hill

Heyuan Yao
Northwestern University

Izzy Grosof
Northwestern University

Benjamin Berg
UNC Chapel Hill

ABSTRACT

Modern data center servers process multiple jobs in parallel to improve performance. However, each job demands some subset of a server's resources (e.g., CPUs, memory, storage), and a set of jobs can run in parallel only if there are sufficient computational resources to meet each job's needs. Given a stream of arriving jobs, a *scheduling policy* must choose a set of jobs to run in parallel at every moment in time. While prior work has studied the stability and mean response time of various scheduling policies in this setting, the vast majority of this work assumes that jobs can be preempted at any time with no overhead. In practice, however, datacenter jobs accumulate significant state as they run, making preemption costly or even impossible. In these limited preemption scenarios, little is known about the optimal way to schedule jobs and avoid excessive preemption overhead.

We consider a model of a datacenter server processing non-preemptible jobs on the fluid scale. We show that, when jobs are non-preemptible, it is difficult to both stabilize the system *and* maintain high utilization of the server resources. We then derive a class of policies that asymptotically minimizes the amount of wasted server resources in some cases. While our policies are optimal when job resource demands are simple, we show how the problem becomes more difficult as job resource demands become more varied. We then discuss potential approaches for deriving optimal policies in these more complex cases.

1. INTRODUCTION

Data center servers process many jobs in parallel. Every job demands a set of computational resources that it requires to run. Hence, a set of jobs can only be run in parallel if the server has sufficient resources to satisfy the needs of each job. We call a set of jobs that can run in parallel a *feasible schedule*. A scheduling policy must choose a feasible schedule to use at every moment in time to both stabilize the system and reduce the mean response time across jobs. What makes this problem particularly difficult is that jobs accumulate a significant amount of state as they run, making preemption impossible. As a result, the scheduler cannot freely transition between different feasible schedules.

To transition from one feasible schedule to another, the system must wait for some currently running jobs to finish. Some queued jobs can then be admitted into service to transition the system to the next feasible schedule. During this process of draining the server, the server resources can be dramatically underutilized. Hence, we refer to the length of this draining period as *switching overhead*.

MAMA 2026 Ann Arbor, Michigan

Prior work on scheduling datacenter jobs with resource demands [2, 3, 4] has studied the conditions for system stability and has designed scheduling policies to reduce the mean response time across jobs. Much of this work assumes jobs are preemptible [4]. Other work proves stability results in the non-preemptive setting [2, 3], but suggests policies that perform poorly in practice due to excessive switching overhead. Only one recent policy [1] directly optimizes for mean response time in the non-preemptive setting, but this work only handles the case where jobs belong to one of two job classes. Our work aims to optimize the mean response time of non-preemptive jobs in a far more general model.

In this paper, we study a fluid model of non-preemptible datacenter jobs. We prove an order-wise tight lower bound on switching overheads in this model. To attain this asymptotic lower bound, we propose a class of policies called *scaling cycles* and formulate a constrained optimization problem to solve for the optimal scaling cycle. While our optimization problem is complex, we obtain a closed-form for the optimal scaling cycle in some cases.

2. OUR FLUID LIMIT MODEL

We consider a server with k types of jobs. We assume that all type-i jobs demand the same amount of server resources. We let $\mathbf{u} = (u_1, \cdots, u_k) \in \mathbb{Z}^k_{\geq 0}$ be a schedule, where u_j is the number of type-j jobs served. If the resource demands of all the jobs in $\mathbf{u}$ can be met by the server, these jobs can run in parallel and we refer to $\mathbf{u}$ as a *feasible schedule*. Let $\mathcal{S} \subseteq \mathbb{Z}^k_{\geq 0}$ be the set of all feasible schedules.

We consider an infinite stream of jobs arriving into the server. We assume that type-j jobs arrive according to a Poisson process with rate λ_j. Let $\boldsymbol{\lambda} = (\lambda_1, \cdots, \lambda_k)$ be the arrival rate vector. We consider the case where jobs are *non-preemptible*. That is, once a job enters service, it must run uninterrupted to completion. We refer to the amount of service a job requires as the job's *size*. We assume that the job sizes of type-j jobs are drawn i.i.d. from a phase-type distribution S_j, where $\mathbb{E}[S_j] = 1/\mu_j$. Let $\boldsymbol{\mu} = (\mu_1, \cdots, \mu_k)$ be the vector of job sizes for each class.

We describe the state of the system as a vector

$$\mathbf{Q}(t) = (Q_1(t), Q_2(t), \cdots, Q_k(t)),$$

where $Q_j(t)$ denotes the number of type-j jobs present at time t. At every time $t \in \mathbb{R}_{\geq 0}$, a scheduling policy must choose a schedule $\mathbf{u}(t) \in \mathcal{S}$.

Let $(\mathbf{u}_1, \mathbf{u}_2, \cdots)$ be a sequence of feasible schedules applied by the scheduling policy, with each schedule $\mathbf{u}_i$ active for a duration $t_i \geq 0$. Let $\mathbf{t} = (t_1, t_2, \cdots)$ denote the sequence for durations. Because jobs are nonpreemptible, the policy can only transition between schedules if doing so does not require interrupting active jobs. Specifically, to change the schedule from $\mathbf{u}_1$ to $\mathbf{u}_2$, the vector of jobs currently in service

must be component-wise at most $\mathbf{u}_2$. Hence, one way to facilitate changing schedules is to stop admitting new jobs into service until the vector of jobs in service falls below the desired schedule. We refer to the time required to switch the schedule to a desired target schedule as *switching overhead*.

We consider the fluid limit of the above system. Specifically, we consider a sequence of systems indexed by r that evolve according to $\mathbf{Q}^{(r)}(t)$. We call this index r the *scaling factor*. Let $\mathbf{Q}^0 \in \mathbb{Z}^k_{\geq 0}$ be an initial state of the system. For a system with scaling factor r, we let the system start with the state $\mathbf{Q}^{(r)}(0) = r\mathbf{Q}^0$. We consider a class of sequences that we call *limited-preemption sequences* where the switching overheads scale slower than r. That is, the switching overheads go to zero when we take fluid limit. Then, by standard techniques from the fluid limit literature [5], we know that this sequence of systems converge to a fluid limit system $\lim_{r\to\infty} r^{-1}\mathbf{Q}^{(r)}(rt) = \bar{\mathbf{Q}}(t)$, where $\bar{\mathbf{Q}}(t)$ denotes the fluid limit queue lengths at time t. We let $\bar{\mathbf{Q}}^i$ be the fluid queue lengths $\bar{\mathbf{Q}}(t)$ evaluated at the time when the i-th schedule in the sequence completes, $\bar{\mathbf{Q}}^i = \bar{\mathbf{Q}}\left(\sum_{n=1}^{i} t_n\right)$. Then, the dynamics on $\bar{\mathbf{Q}}^i$ can be written as

$$\forall i = 1, 2, \cdots, \bar{\mathbf{Q}}^i = \left(\bar{\mathbf{Q}}^{i-1} + t_i(\boldsymbol{\lambda} - \boldsymbol{\mu} \odot \mathbf{u}_i)\right)^+,$$

where $\odot$ is the component-wise vector multiplication, and $(\mathbf{x})^+ = \max(\mathbf{x}, 0)$. Our goal is to find the shortest sequence of feasible schedules and the corresponding durations, such that we can empty the queue lengths at the fluid scale in the shortest possible time. We consider the queue empty at time t, if $\bar{\mathbf{Q}}(t) \prec \epsilon$ for some $\epsilon > 0$.

Given initial queue lengths $\mathbf{Q}^0 \in \mathbb{R}^k_{\geq 0}$, picking a feasible schedule $\mathbf{u}$ produces a *drift* $\boldsymbol{\lambda} - \boldsymbol{\mu} \odot \mathbf{u}$ on the queue lengths, moving the system for some duration t_1 to a new fluid state $\bar{\mathbf{Q}}^1$. Crucially, to maintain high server utilization, we enforce a "no waste" constraint: the scheduler cannot continue to apply a negative drift to a job type whose queue is already empty. Mathematically, this means the fluid trajectory must remain naturally non-negative, allowing us to drop the $(\mathbf{x})^+$ operator and strictly require $\bar{\mathbf{Q}}^i = \mathbf{Q}^{i-1} + t_i(\boldsymbol{\lambda} - \boldsymbol{\mu} \odot \mathbf{u_i}) \succeq \mathbf{0}$. *Our primary objective is to empty the fluid queue lengths in the shortest possible time, subject to strictly avoiding wasted resources and using a limited-preemption sequence.*

The set of feasible schedules, $\mathcal{S}$, can grow exponentially with the number of job types, k. Fortunately, prior work [2] shows how to derive a small subset of schedules $\mathcal{C} \subseteq \mathcal{S}$, where $|\mathcal{C}| \leq k$, that can both stabilize the system and achieve good mean response times when jobs are preemptible. Hence, we will design policies that use the feasible schedules in $\mathcal{C}$.

Let schedules in the subset be $\mathcal{C} = \{\mathbf{x}_1, \cdots, \mathbf{x}_\ell\}$, where $\ell = |\mathcal{C}|$. We define the drift of each schedule as $\mathbf{d}_i = \boldsymbol{\lambda} - \boldsymbol{\mu} \odot \mathbf{x}_i$, where $\mathbf{d}_i = (d_{i1}, d_{i2}, \cdots, d_{ik}) \in \mathbb{R}^k$ for all $i \in [1, \ell]$. For notational convenience, we let $\mathbf{d} = (\mathbf{d}_1, \cdots, \mathbf{d}_\ell)$ denote the vector of these drifts and let $D = [d_{ij}]_{i=1..\ell, j=1..k} \in \mathbb{R}_{\ell \times k}$ be the corresponding drift matrix.

With these drifts defined, we can formally express the stability of the system using only the schedules in $\mathcal{C}$: stability implies that there exists a strictly positive weight vector $\boldsymbol{\alpha} \in \mathbb{R}^\ell_{>0}$ such that $\langle \boldsymbol{\alpha}, \mathbf{d} \rangle \prec \mathbf{0}$.

In this fluid model, the system's dynamics are entirely driven by these ℓ drift vectors. If $\ell < k$, the cycle is confined to an ℓ-dimensional subspace, meaning we can project the state space down from $\mathbb{R}^k$ to $\mathbb{R}^\ell$. Therefore, without loss of generality, we assume $\ell = k$. Furthermore, we assume that the system is *complete*, in the way that all drifts are needed to stabilize the system. Formally, a complete system implies that every vector in the closed negative quadrant $\mathbf{q} \in \mathbb{R}^k_{\leq 0}$ can be represented as a strictly positive linear combination of the drifts, $\mathbf{q} = \langle \boldsymbol{\alpha}, \mathbf{d} \rangle$ for some $\boldsymbol{\alpha} \in \mathbb{R}^k_{>0}$. This ensures all k schedules are required to stabilize the system, or to achieve any non-positive drift.

Recall that our primary objective is to empty the system in the shortest possible time using a *limited-preemption sequence*. This requires the total number of schedule switches to scale strictly sublinearly with respect to the initial queue size. Before designing a specific policy to achieve this, we must first understand the fundamental limits of the system to see if such a sequence is even possible: what is the absolute minimum number of schedule switches required to empty the fluid queues?

The following lemma provides this theoretical lower bound.

LEMMA 1. *In a complete system where every drift has at least one positive element, given initial queue lengths $\mathbf{Q}^0 \in \mathbb{R}^k_{\geq 0}$, the number of schedule switches required to empty the fluid queue lengths is lower bounded by $\Omega(\log \|\mathbf{Q}^0\|_1)$.*

PROOF. The scaling factor of the cycle is lower bounded by the maximum of the ratio between the positive element and the most negative element in each drift. Hence, we need at least $\Omega(\log \|\mathbf{Q}^0\|_1)$ cycles to empty the fluid queues. □

To achieve this asymptotically optimal lower bound, we propose a policy that runs schedules in $\mathcal{C}$ cyclically according to a *service order permutation* $\pi : \{1, \cdots, k\} \to \{1, \cdots, k\}$. That is, we execute the sequence $(\mathbf{u}_1, \mathbf{u}_2, \cdots)$, where $\mathbf{u}_i = \mathbf{x}_{\pi(n)}$ if $i \equiv n \pmod{k}$.

Our mathematical objective is to design this cycle such that the queue lengths contract uniformly in every direction after one complete cycle of k schedules. That is, we seek a sequence such that $\bar{\mathbf{Q}}^k = c\mathbf{Q}^0$ for some *scaling constant* $0 < c < 1$. If such a uniform geometric contraction exists, emptying the queue requires exactly $O(\log \|\mathbf{Q}^0\|_1)$ cycle iterations. Because this exactly matches the theoretical lower bound from Lemma 1, this policy asymptotically minimizes the number of preemptions. We refer to cycles that achieve this property as *scaling cycles*.

3. OPTIMAL SCALING CYCLES

To solve for the optimal scaling cycle, we must jointly determine the duration vector $\mathbf{t}$, the initial state $\mathbf{Q}^0$, and the scaling factor c. It is computationally expensive to search over the space of all possible service order permutations, so we consider a fixed permutation π in this section. We now show that the initial state $\mathbf{Q}^0$ can be entirely derived from the duration vector and the scaling factor, meaning it is sufficient to optimize only over $\mathbf{t}$ and c.

At the end of the cycle, the fluid lengths $\bar{\mathbf{Q}}^k$ can be computed by accumulating the durations applied to each drift:

$$\bar{\mathbf{Q}}^k = \mathbf{Q}^0 + \sum_{i=1}^{k} t_i \mathbf{d}_{\pi(i)}.$$

Substituting the uniform contraction requirement $\bar{\mathbf{Q}}^k = c\mathbf{Q}^0$ and rearranging, we get

$$\mathbf{Q}^0 = (c-1)^{-1} \sum_{i=1}^{k} t_i \mathbf{d}_{\pi(i)}.$$

This confirms that $\mathbf{Q}^0$ is strictly a function of $\mathbf{t}$ and c. Moreover, because the fluid model is linear, the optimal scaling constant c^* depends only on the relative direction of the trajectory, not the absolute magnitude of $\mathbf{Q}^0$. Therefore, we can safely eliminate a redundant degree of freedom by normalizing the total time spent in one cycle. We restrict our search to duration vectors where $\|\mathbf{t}\|_1 = 1$.

We can then find the optimal scaling cycle by searching for the normalized durations that minimize c, subject to the strict constraint that no intermediate queue length ever drops below zero to avoid wasting resources.

THEOREM 1. *In a complete system, for a given service permutation order π, we can solve the following constrained optimization problem to construct the optimal scaling cycle:*

$$\arg\min_{c,t_1,\ldots,t_k} c$$
$$s.t. \begin{cases} \bar{Q}_j^i = -c\sum_{n=1}^{i} t_n d_{\pi(n)j} - \sum_{n=i+1}^{k} t_n d_{\pi(n)j} \geq 0, \\ 1 \leq i, j \leq k, \\ t_1 + \cdots + t_k = 1. \end{cases} \quad (1)$$

PROOF. This follows directly from no intermediate queue lengths drop below zero and the normalization on $\mathbf{t}$. □

To minimize the scaling factor c, the cycle trajectory must be pushed to its limits against these non-negativity constraints. If the cycle never completely emptied any queue, the scaling factor could be decreased further without violating the constraints. Therefore, the optimal cycle must intentionally empty one of the fluid queues. Mathematically, this means the trajectory must hit the boundary of the positive orthant $\mathbb{R}^k_{\geq 0}$, activating at least one inequality constraint such that $\bar{Q}_j^i(\mathbf{t}^*, c^*) = 0$.

The following lemma formalizes this boundary-contacting behavior, demonstrating that the optimal trajectory is tightly constrained by the boundaries of $\mathbb{R}^k_{\geq 0}$.

LEMMA 2. *In a complete system, the optimal scaling cycle must contact the boundaries of $\mathbb{R}^k_{\geq 0}$ at least k times. Formally, at least k of the nonnegative constraints must be active in (1) for solution $(\mathbf{t}^*, c^*)$, meaning that $\bar{Q}_j^i(\mathbf{t}^*, c^*) = 0$ for at least k pairs of (i, j).*

Special case: $\mathbf{k \leq 3}$. Lemma 2 shows that the optimal cycle hits the boundaries of $\mathbb{R}^k_{\geq 0}$ at least k times. When $k \leq 3$, we can strengthen this result in the following theorem:

THEOREM 2. *In a complete system where $k \leq 3$, the optimal scaling cycle contacts the boundary of every job type at least once. Specifically, for every job type j, there exists some intermediate step i such that $\bar{Q}_j^i = 0$.*

PROOF SKETCH. Follows directly from Lemma 2 and the linear independence among k drifts. □

Under certain conditions, the optimal scaling cycle empties one distinct job type at each step. We refer to this as an *all-boundary cycle*. Formally, an all-boundary cycle admits a *target type permutation* $\sigma : \{1, \ldots, k\} \to \{1, \ldots, k\}$ such that $\bar{\mathbf{Q}}^i_{\sigma(i)} = 0$ for each step i. A necessary condition for all-boundary cycles is that a positive drift needs to follow an emptied fluid queue to allow no waste. We call (π, σ) a *valid permutation pair* if $d_{\pi(i),\sigma(i)} < 0$ and $d_{\pi(i+1),\sigma(i)} > 0$ for all $i \in \{1, \ldots, k\}$ (with indices wrapping modulo k).

A major advantage of an all-boundary cycle is its analytical tractability. Because the all-boundary cycle contacts boundaries in a predictable sequence, $\mathbf{Q}^0$ can be related to $\bar{\mathbf{Q}}^k$ via a linear transformation. This allows us to bypass the constrained optimization problem entirely and find the optimal scaling constant c by finding eigenvalues.

THEOREM 3. *If there exists a valid permutation pair (π, σ) in the drift matrix D, we can repermute D according to (π, σ) as $D_{\pi,\sigma}$ to write this cycle as a linear transformation*

$$\bar{\mathbf{Q}}^k = -UL^{-1}\mathbf{Q}^0,$$

where L and U are the lower-triangular and strictly upper-triangular components of $D^T_{\pi,\sigma}$. Consequently, the optimal scaling constant c and initial state $\mathbf{Q}^0$ are an eigenvalue-eigenvector pair of the transition matrix $-UL^{-1}$.

Unfortunately, a valid permutation pair is no longer sufficient to guarantee an all-boundary cycle when $k \geq 4$.

Conjectures for General k. Our investigation into general k-dimensional systems initially focused on generalizing the valid permutation pair structure to find all-boundary cycles. However, we found counterexamples when $k \geq 4$ where collateral cross-drifts prevented the system from emptying exactly one distinct fluid queue per step. Hence, we resort to generalizing Theorem 2 in the following conjecture:

CONJECTURE 1. *Theorem 2 holds for $k \geq 4$.*

While a formal proof for Conjecture 1 remains part of our ongoing work, we outline a potential theoretical approach. We believe this property can be established by analyzing the Karush-Kuhn-Tucker (KKT) conditions of the optimization program in (1). Specifically, by examining the gradient of the scaling factor c with respect to small local perturbations in the duration vector $\mathbf{t}$, one could show that if a job type's boundary is strictly avoided throughout the entire cycle, there must exist a feasible perturbation that further reduces c, thereby contradicting optimality.

4. CONCLUSION

We consider the problem of scheduling non-preemptible datacenter jobs and develop a fluid model of this setting. We devise a class of scheduling policies, called scaling cycles, that asymptotically minimize switching overhead in the fluid model. We show that the difficulty of finding an optimal scaling cycle depends on the number of job types, k.

5. REFERENCES

[1] CHEN, Z., ANGGRAITO, A., OLLIARO, D., MARIN, A., MARSAN, M. A., BERG, B., AND GROSOF, I. Improving nonpreemptive multiserver job scheduling with quickswap. *PEVA 171* (2026), 102525.

[2] CHEN, Z., GROSOF, I., AND BERG, B. Improving multiresource job scheduling with markovian service rate policies. *POMACS 9*, 2 (June 2025).

[3] GHADERI, J. Randomized algorithms for scheduling vms in the cloud. In *INFOCOM* (2016), IEEE, pp. 1–9.

[4] STOLYAR, A. L. Maxweight scheduling in a generalized switch: State space collapse and workload minimization in heavy traffic. *The Annals of Applied Probability 14*, 1 (2004), 1–53.

[5] WHITT, W. *Stochastic-process limits: an introduction to stochastic-process limits and their application to queues*. Springer, 2002.