Improving Nonpreemptive Multiserver Job Scheduling with Quickswap

Zhongrui Chen [*]
jcpwfloi@cs.unc.edu

Adityo Anggraito [†]
adityo.anggraito@unive.it

Diletta Olliaro [†]
diletta.olliaro@unive.it

Andrea Marin [†]
marin@unive.it

Marco Ajmone Marsan [‡]
marco.ajmone@imdea.org

Benjamin Berg [*]
ben@cs.unc.edu

Isaac Grosof [§]
izzy.grosof@northwestern.edu

ABSTRACT

Modern data center workloads are composed of *multiserver jobs* that require multiple servers in order to run. Many multiserver jobs can run in parallel as long as there are enough servers to satisfy the demand of each job. Given a stream of arriving jobs, a *scheduling policy* must determine which jobs to run at every moment in time to minimize the mean response time across jobs — the average time from when a job arrives to the system until it is complete. Because preemptions are often computationally expensive in data centers, the scheduling policy cannot preempt running jobs.

We propose a new class of non-preemptive policies called *Most Servers First with Quickswap* (MSFQ). MSFQ policies prioritize jobs that demand more servers. However, they keep mean response time low by periodically granting priority to other jobs in the system. We prove conditions under which MSFQ policies stabilize the system, and analyze their mean response time. We prove that the MSFQ priority switching mechanism allows these policies to greatly outperform state-of-the-art policies that rely on simpler priority mechanisms. We validate our theoretical results via extensive simulations using traces from the Google Borg system.

1. INTRODUCTION

Modern data centers serve *multiserver jobs* that occupy multiple servers simultaneously. Each multiserver job is associated with a *server need*, the number of servers the job requires to run, and a *service duration*, the amount of time the job must run to be completed. A set of multiserver jobs can run in parallel only if the system has enough dedicated servers for each job. A *scheduling policy* must select which jobs to run in parallel at every moment in time. Given a fixed number of servers, k, our goal is to choose a scheduling policy that minimizes the *mean response time* across jobs in a stream of arriving multiserver jobs — the average time from when a job arrives to the system until it is completed.

[*]University of North Carolina at Chapel Hill, USA
[†]Università Ca' Foscari Venezia, Italy
[‡]IMDEA Networks Institute, Spain
[§]Northwestern University, USA

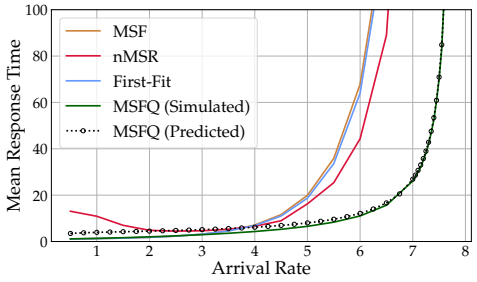

Figure 1: Mean response times as a function of arrival rates for MSFQ and competitor policies.

Designing scheduling policies for multiserver jobs is difficult for two reasons: (i) it is hard to utilize all servers in the system due to bin-packing effects; (ii) modern multiserver jobs are *stateful*, making preemptions costly or impossible. This raises the question of how to design and analyze nonpreemptive scheduling policies that minimize the mean response time across a stream of multiserver jobs.

To achieve high server utilization, prior work on scheduling multiserver jobs has suggested prioritizing jobs with larger server needs. Specifically, [3] evaluated the Most-Servers-First (MSF) policy, a non-preemptive policy that prioritizes jobs with larger server needs. MSF considers all jobs in descending server need order. If there are enough servers to satisfy a job, MSF puts this job into service.

To understand the pros and cons of MSF, consider an example where jobs either need one server or k servers. We refer to this as the *one-or-all* setting. Here, MSF serves jobs in two alternating *phases*: (i) MSF serves k-server jobs until none remain in the system; (ii) MSF serves 1-server jobs until none remain before returning to serve k-server jobs. We show in our full paper [1] that MSF achieves optimal long-run average server utilization in the one-or-all setting. Unfortunately, this high server utilization does not lead to low mean response time. MSF suffers because, as the job arrival rate increases, it takes an increasingly long time to switch between phases. This creates a feedback loop where 1-server jobs accumulate while the system processes k-server jobs, leading to a long phase of serving 1-server jobs during which many k-server jobs accumulate.

To address the slow phase switching of MSF, we develop a class of scheduling policies for the one-or-all setting called

Most Server First with Quickswap (MSFQ). We show that MSFQ policies stabilize the system whenever it is possible to do so (Theorem 1), and we analyze the mean response time of MSFQ (Theorem 2). To evaluate MSFQ policies, we compare them against MSF, First-Fit (an optimized first-come-first-served policy), and a non-preemptive Markovian Service Rate (nMSR) policy [2] (see Figure 1). We also develop variants of MSFQ that process general multiserver job workloads, and evaluate these variants using traces from Google Borg [4] (see Figure 2).

2. OUR MODEL

We consider a system with k servers. A multiserver job is represented by an ordered pair (i, s), where $i \in \{1, 2, \cdots, k\}$ is the number of servers the job needs to run and s is the job's *service duration*, the time the job must run on the servers before completion. Jobs occupy a fixed number of servers throughout their time in service, and a job in service cannot be preempted. We refer to this job model as the Multiserver Job (MSJ) model.

We consider workloads composed of several *job classes*, where class-i jobs each require i servers. We consider serving a stream of multiserver jobs, where class-i jobs arrive according to an independent Poisson process with rate λ_i. We further assume the service durations of class-i jobs are i.i.d. exponentially distributed random variables such that $S_i \sim \exp(\mu_i)$.

A set of jobs can run in parallel if their aggregate server demand does not exceed the number of available servers. That is, let $\mathbf{u} = (u_1, u_2, \cdots, u_k)$ denote a set of multiserver jobs containing u_i class-i jobs. These jobs can run in parallel if and only if $\sum_{i=1}^{k} i\, u_i \leq k$.

3. MAIN RESULTS

We introduce the class of Most Server First with Quickswap (MSFQ) policies in the one-or-all setting. MSFQ prioritizes jobs with high server needs, following similar service phases as the MSF policy. However, to shorten the durations of each phase, an MSFQ policy is associated with a threshold, ℓ. Instead of serving class-1 jobs in phase 2 until none remain, MSFQ serves class-1 jobs until there are ℓ jobs remaining in the system. At this point, MSFQ stops admitting new class-1 jobs into service and completes the ℓ class-1 jobs in service. Once these jobs are complete, MSFQ begins serving class-k jobs.

We show that, by using the threshold ℓ to abbreviate phase 2, MSFQ policies can reduce the feedback loop that leads to high mean response time under MSF. We first show that MSFQ policies stabilize the system whenever it is possible to do so. That is, if it is possible to achieve a finite mean response time, any MSFQ policy will have a finite mean response time.

THEOREM 1. *A Most Servers First with Quickswap (MSFQ) policy with any threshold ℓ, $0 \leq \ell < k$, stabilizes the system whenever it is possible to do so in the one-or-all system.*

PROOF. We first show that no scheduling policy is stable if $\lambda_1/k\mu_1 + \lambda_k/\mu_k \geq 1$ by comparing to a resource-pooled system. Next, we show that our MSFQ policy is stable given that $\lambda_1/k\mu_1 + \lambda_k/\mu_k < 1$ using a Lyapunov drift argument. □

Next, we analyze the mean response time of an MSFQ policy under the one-or-all setting.

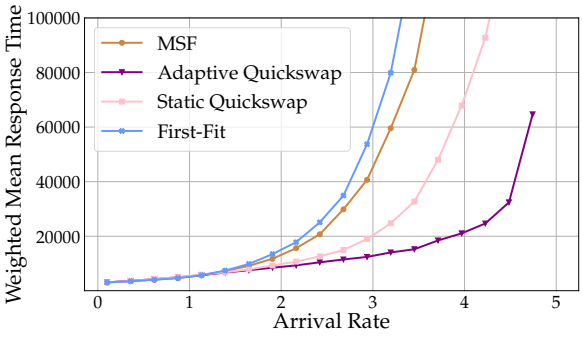

Figure 2: Mean response times as a function of arrival rates for MSFQ and competitor policies using Google Borg traces.

THEOREM 2. *The mean response time under an MSFQ policy, $\mathbb{E}[T]$, depends on the first and second moments of the durations of each phase and the mean number of jobs at the beginning of each phase. For a given MSFQ policy, we can compute all of these quantities.*

PROOF. We show that $\mathbb{E}[T]$ can be written in terms of phase durations by conditioning on which phase a job arrives during, and whether a job is a class-1 or a class-k job.

We then derive the Laplace transforms of the phase durations and the expected number of jobs at the beginning of each phase. We use these transforms to compute the overall mean response time. Although the transforms we derive are recursively defined, we provide a calculator[1] that symbolically differentiates these transforms to compute the mean response time. □

4. EVALUATIONS

We evaluate the accuracy of our mean response time analysis and compare MSFQ with the MSF, nMSR and First-Fit policies via simulations of the one-or-all setting. Figure 1 shows that our analysis is accurate and that MSFQ outperforms all competitor policies at all arrival rates.

We also develop variants of MSFQ called Adaptive Quickswap and Static Quickswap that work outside of the simplified one-or-all setting. For this evaluation, we use simulations driven by traces from the Google Borg cluster scheduler. Here, 86% of the system load is contributed by the 0.34% of jobs with the largest server needs. To ensure that our scheduling policy does not ignore these important jobs, we evaluate this trace using a weighted mean response time metric that weights class-i jobs by the amount of load they contribute to the system. We show that our policies perform favorably compared to all competitors on this weighted mean response time metric (see Figure 2).

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. *Performance Evaluation* (2025).

[2] CHEN, Z., GROSOF, I., AND BERG, B. Improving multiresource job scheduling with markovian service rate policies. *POMACS 9*, 2 (2025), 1–36.

[3] SPEITKAMP, B., AND BICHLER, M. A Mathematical Programming Approach for Server Consolidation Problems in Virtualized Data Centers. *IEEE Transactions Services Computing 3*, 4 (2010), 266–278.

[1] https://github.com/jcpwfloi/msfq-calculator

[4] Tirmazi, M., Barker, A., Deng, N., Haque, M. E., Qin, Z. G., Hand, S., Harchol-Balter, M., and Wilkes, J. Borg: the next generation. In *EuroSys* (2020), pp. 1–14.