Simple Policies for Multiresource Job Scheduling

Zhongrui Chen
University of North Carolina at Chapel Hill

Isaac Grosof
University of Illinois, Urbana-Champaign

Benjamin Berg*
University of North Carolina at Chapel Hill

ABSTRACT

Data center workloads are composed of *multiresource jobs* requiring a variety of computational resources including CPU cores, memory, disk space, and hardware accelerators. Modern servers can run multiple jobs in parallel, but a set of jobs can only run in parallel if the server has sufficient resources to satisfy the demands of each job. It is generally hard to find sets of jobs that perfectly utilize all server resources, and choosing the wrong set of jobs can lead to low resource utilization. This raises the question of how to allocate resources across a stream of arriving multiresource jobs to minimize the *mean response time* across jobs — the mean time from when a job arrives to the system until it is complete.

Current policies for scheduling multiresource jobs are complex to analyze and hard to implement. We propose a class of *simple* policies, called *Markovian Service Rate* (MSR) policies. We show that the class of MSR policies is *throughput-optimal*, in that if a policy exists that can stabilize the system, then an MSR policy exists that stabilizes the system. We derive bounds on the mean response time under an MSR policy, and show how our bounds can be used to choose an MSR policy that minimizes mean response time.

1. INTRODUCTION

Modern data centers serve *multiresource jobs* that require a variety of computational resources. Different jobs request different amounts of each resource. For example, CPU-intensive jobs may request little memory, and memory-intensive jobs may require only a few CPU cores. A data center server can serve multiple jobs in parallel, but has a limited amount of each resource. Hence, a set of jobs can only run in parallel if the server can satisfy the resource demands of each job. This raises the question of what set of jobs to run on the server at every moment in time. Specifically, we consider scheduling a stream of multiresource jobs to minimize the mean response time across jobs — the average time from when a job arrives to the system until it is completed.

Unfortunately, due to bin-packing effects, it is generally impossible to guarantee that all server resources will be utilized. For example, consider a server with 20 CPUs and 20 GB of memory that processes some jobs that require 10 CPUs and 4 GB of memory, and others that require 4 CPUs and 10 GB of memory. No combination of these jobs will simultaneously utilize all 20 cores and all 20 GB of memory. Given that some resources must go unused, it is unclear which imperfect allocations will minimize the mean response time. Our goal is to devise a *scheduling policy* that chooses which jobs to run in parallel at every moment in time.

Prior Work. The well-known results of [6] analyze a policy called *Max-Weight* that repeatedly solves a weighted knapsack problem to decide what set of multiresource jobs to serve in parallel. Max-Weight is shown to be *throughput-optimal*: if a scheduling policy exists that provides finite mean response time, Max-Weight provides finite mean response time. Furthermore, [1] shows that Max-Weight is asymptotically optimal in a conventional heavy-traffic limit. Unfortunately, Max-Weight involves repeatedly solving a computationally hard problem in order to find good allocation decisions, limiting the practicality of the policy.

While less complex policies for scheduling multiresource jobs have been analyzed, their lower complexity comes with a cost. Specifically, the First-Come-First-Served (FCFS) policy was recently analyzed in [4]. While this work derives bounds on the mean response time of FCFS, FCFS is far from optimal and can cause instability in cases where Max-Weight performs well. An easy-to-implement, throughput-optimal policy is presented in [3, 7], but there is no known analysis of the mean response time for this policy.

Our Results. We analyze the simple class of *Markovian Service Rate* (MSR) scheduling policies. This class of policies is throughput-optimal: if it is possible to stabilize the system, there exists an MSR policy that stabilizes the system. We bound the mean response time under an MSR policy, and use our bounds to pick the MSR policy that minimizes mean response time. Our results fill a gap in the literature on multiresource jobs: MSR policies are simpler to implement and analyze than Max-Weight, but unlike FCFS, the class of MSR policies is throughput-optimal.

2. MULTIRESOURCE JOB MODEL

We consider a server with R different types of computational resources. Let $\mathbf{C} = (C_1, C_2, \cdots, C_R) \in \mathbb{R}_+^R$ denote the resource capacity of the server, where C_j is the amount of the j-th resource the server has. A multiresource job can be represented by a *resource demand vector*, the amount of each resource needed to run, and a *service requirement*, the amount of time the job must run on the server before completion. Multiple multiresource jobs can run on the server in parallel as long as the cumulative resource demand of the jobs doesn't exceed the server capacity. We call this restriction the *resource constraint*. We assume that there are K types of jobs, where type-i jobs share a resource demand vector $\mathbf{J}^i = (J_1^i, J_2^i, \cdots, J_R^i) \in \mathbb{R}_+^R$ and the service requirements of type-i jobs are sampled i.i.d. from a common distribution $S_i \sim \exp(\mu_i)$. We assume that the scheduler knows the resource demand and service requirement distribution of each job, but the exact service requirement of each job is unknown. While our analysis can be adapted to handle phase-

[*] Suppported by National Science Foundation grants NSF-CCF-2403195 and NSF-IIS-2322974

MAMA 2024 Venice, Italy

type service requirement distributions, we limit ourselves to exponential distributions here due to space constraints.

We define a *possible schedule* as a set of jobs that can run in parallel without violating the resource constraints of the server. We describe a possible schedule using a vector $\mathbf{u} = (u_1, u_2, \cdots, u_K) \in \mathbb{Z}_+^K$, where u_i denotes the number of type-i jobs being served. The resource constraint in this case translates to $\sum_{i=1}^K u_i \mathbf{J}^i \leq \mathbf{C}$. We call the set of all possible schedules the *schedulable set*, $\mathcal{S}$. A *scheduling policy* in the multiresource job model, $\boldsymbol{u}(t)$, chooses a schedule from $\mathcal{S}$ to use at time t. Note that $\boldsymbol{u}(t)$ need not depend on time and can depend on external states such as jobs in the system.

We assume that type-i jobs arrive to the system according to a Poisson process with rate λ_i. Let

$$\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \cdots, \lambda_K) \in \mathbb{R}_+^K$$

denote the vector of arrival rates. Similarly, we let

$$\boldsymbol{\mu} = (\mu_1, \mu_2, \cdots, \mu_K) \in \mathbb{R}_+^K$$

denote the vector of service rates. We describe the state of the system as the vector

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

where $Q_i(t)$ is the number of type-i jobs present at time t.

Note that $\{\mathbf{Q}(t), t \geq 0\}$ is a Continuous-Time-Markov-Chain (CTMC). We are interested in analyzing the steady-state behavior of the system as $t \to \infty$. Specifically, let $\mathbf{Q} \sim \lim_{t\to\infty} \mathbf{Q}(t)$. We will analyze the mean response time across type-i jobs, $\mathbb{E}[T_i]$. To bound $\mathbb{E}[T_i]$, we will bound $\mathbb{E}[Q_i]$ and apply Little's Law. Let $\mathbf{T} = (T_1, T_2, \cdots, T_K)$ be a vector of random variables that denotes the response times of type-i jobs. We have $\mathbb{E}[\mathbf{T}] = \mathbb{E}[\mathbf{Q}]/\boldsymbol{\lambda}$. We say the system is *stable* if the CTMC $\{\mathbf{Q}(t)\}$ is positive recurrent.

3. MARKOVIAN SERVICE RATE POLICIES

Max-Weight's complexity stems from repeatedly choosing schedules from the entire set of possible schedules, $\mathcal{S}$. Additionally, the rule for selecting a schedule is fairly opaque: each choice requires solving a hard optimization problem. This complexity obscures the intuitive property that leads Max-Weight to stabilize the system. Namely, Max-Weight guarantees that the average completion rate of type-i jobs is greater than the arrival rate of type-i jobs. We will derive policies that achieve this property via a one-time, offline optimization step rather than repeated online optimization.

We define a class of scheduling policies called *Markovian Service Rate* (MSR) policies. An MSR policy, π, chooses a set $\mathcal{S}^\pi \subseteq \mathcal{S}$ of possible schedules called the *candidate set*. The policy then uses a CTMC, $\{\pi(t), t \geq 0\}$, to modulate among the schedules in the candidate set. We call $\{\pi(t)\}$ the *modulating process*. Formally, let $|\mathcal{S}^\pi| = L_\pi$, $\mathcal{S}^\pi = \{\mathbf{u}^1, \mathbf{u}^2, \cdots, \mathbf{u}^{L_\pi}\}$, and $\pi : \mathbb{R}_+ \to \{1, 2, \cdots, L_\pi\}$. The schedule used by π at time t is defined as $\mathbf{u}(t) = \mathbf{u}^{\pi(t)}$. Note that $\mathbf{u}^{\pi(t)} \in \mathcal{S}^\pi \subseteq \mathcal{S}$, so that the schedule chosen at any time t satisfies the server's resource constraints. Note also that $\pi(t)$ is oblivious to the state of the system $\mathbf{Q}(t)$. Let $\mathbb{E}[\mathbf{u}^\pi]$ denote the time average schedule used by π.

We first prove Lemma 1, which provides a sufficient condition for the stability of the system under an MSR policy.

LEMMA 1. *If $\boldsymbol{\lambda} < \mathbb{E}[\mathbf{u}^\pi] \cdot \boldsymbol{\mu}$, then the system is stable, where $\cdot$ and $<$ are element-wise.*

PROOF. We invoke the Foster-Lyapunov theorem [5] with the test function $\|\mathbf{Q}\|^2$ to show that the system is stable. □

We now prove that the class of MSR policies is throughput-optimal. We also show that an MSR policy requires only K candidate schedules to stabilize the system. Hence, even if $\mathcal{S}$ is large, a good MSR policy can be computed efficiently.

THEOREM 1. *Given a system with K job types and an arrival vector $\boldsymbol{\lambda}$, if there exists a scheduling policy that stabilizes the system, there also exists an MSR policy, π, with $L_\pi = K$ that stabilizes the system.*

PROOF. The system can only be stabilized if there exists a weighted average of schedules such that the average completion rate of each job type is larger than its arrival rate [6]. Using Carathéodory's theorem [2], we additionally see that when the system can be stabilized, there exists a weighted average of at most K schedules such that the average completion rate of each job type dominates its arrival rate. We choose $\mathcal{S}^\pi$ to be this set of K points and set the modulating process $\{\pi(t)\}$ such that the average completion rate of each job type is greater than its arrival rate. Finally, Lemma 1 tells us that the system will be stable under π. □

Response Time Bounds on MSR Policies. Theorem 1 tells us a stable MSR policy exists, but such policy may not be unique. Unfortunately, Theorem 1 provides no hints about how to choose from all stable MSR policies. Hence, we derive bounds on the mean response time of an arbitrary MSR policy, and show how these bounds can be used to select an MSR policy that minimizes mean response time.

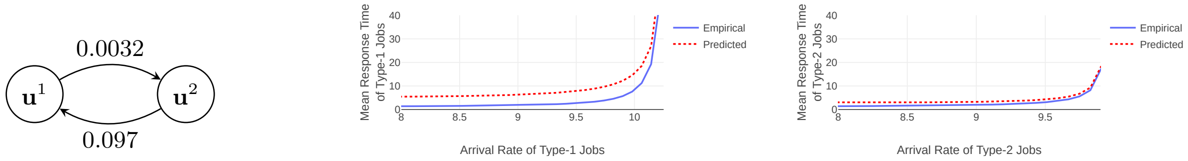
We separately analyze the number of type-i jobs in the system for each job type i. This allows us to compute the mean response time of type-i jobs under an MSR policy π, $\mathbb{E}[T_i^\pi]$. We compare the mean number of type-i jobs in the system under the policy π to the number of jobs in an analogous $M/M/1$ Markovian Service Rate (MSR-1) system [4]. An MSR-1 system under policy $\pi^{MSR\text{-}1}$ consists of a variable-speed server serving jobs that arrive according to a Poisson process with rate $\lambda^{MSR\text{-}1}$. The server has $L_\pi^{MSR\text{-}1}$ states corresponding to different server speeds. Specifically, the server speed is chosen from the vector $(\mu_1^{MSR\text{-}1}, \mu_2^{MSR\text{-}1}, \cdots, \mu_{L_\pi^{MSR\text{-}1}}^{MSR\text{-}1})$ according to a CTMC, $\{\pi^{MSR\text{-}1}(t), t \geq 0\}$. Let $\pi^{MSR\text{-}1}$ be the limiting distribution of the modulating process. We denote the time-average server speed as μ^*. Unlike our original system, the MSR-1 system runs one job at a time and changes server speeds rather than modulating the number of jobs in service. To obtain response time bounds for MSR policies, we compare our original system to an analogous MSR-1 system and then analyze the mean response time of the MSR-1 system.

Consider any job type $i \in [1, K]$. To analyze $\mathbb{E}[T_i^\pi]$, we construct an analogous MSR-1 system, $\pi^{MSR\text{-}1}$, with $L_\pi^{MSR\text{-}1} = L_\pi$. We set $\mu_j^{MSR\text{-}1}$ to be $\mu_i \cdot \mathbf{u}_i^j$. We set the modulating processes, $\{\pi(t)\}$ and $\{\pi^{MSR\text{-}1}(t)\}$, to be the same. We set $\lambda^{MSR\text{-}1} = \lambda_i$. In short, we construct an MSR-1 system that matches the behavior of type-i jobs under π, except the MSR-1 system runs one job at a time. Let $Q_i^{MSR\text{-}1}(t)$ be the number of jobs in the MSR-1 system at time t. We compare these analogous systems in Theorem 2.

THEOREM 2.

$$\mathbb{E}[Q_i^{MSR\text{-}1}] \leq \mathbb{E}[Q_i] \leq \mathbb{E}[Q_i^{MSR\text{-}1}] + \max_j \left\{\mathbf{u}_i^j\right\}$$

PROOF. First, we show that $Q_i^{MSR\text{-}1}(t) \leq_{st} Q_i(t)$ via a coupling argument. Essentially, the service rate in the MSR-1 system is at least that of the original system, because the



(a) Modulating process (b) Type-1 jobs (c) Type-2 jobs

Figure 1: Response time bounds versus simulation for an MSR policy, π. This system serves $K = 2$ job types using $L_\pi = 2$ states, $\mathbf{u}^1 = (10, 10)$ and $\mathbf{u}^2 = (19, 9)$, and the modulating process shown in (a). Here, $\mu_1 = \mu_2 = 1$.

original system may not have enough type-i jobs to run $\boldsymbol{u}_i(t)$ jobs in parallel. The MSR-1 system always serves jobs at rate $\boldsymbol{u}_i(t) \cdot \mu_i$. This implies the lower bound in our claim.

We also prove the upper bound in our claim by a coupling argument. We note that at any time t, we are in one of two cases. First, when $Q_i(t) \leq \max_j \left\{\mathbf{u}_i^j\right\}$, we trivially have

$$Q_i(t) \leq Q_i^{MSR\text{-}1}(t) + \max_j \left\{\mathbf{u}_i^j\right\}.$$

Otherwise, during periods when $Q_i(t) > \max_j \left\{\mathbf{u}_i^j\right\}$, the two systems complete jobs at the same rate. Thus, if our claim holds at the beginning of such a period, it will hold for the entire period. This coupling gives the desired mean response time an upper bound. □

To apply Theorem 2, we can bound the mean response time of any MSR-1 system using the techniques described in [4] to obtain response time bounds for any MSR policy. Corollary 1 gives an example of our response time bounds in a system with two resource types and two job types. Here, we consider an MSR policy that uses two schedules. Let $\alpha_{j,k}$ represent the transition rate of the modulating process from state j to state k in both the original system and the MSR-1 system.

Corollary 1. *In a system with $R = 2$ resources and $K = 2$ job types, we consider an MSR policy, π, with $L_\pi = 2$. WLOG, we assume $\mu_1^{MSR\text{-}1} \leq \mu_2^{MSR\text{-}1}$. In this case,*

$$\mathbb{E}[T_i^\pi] = \frac{1}{\mu^* - \lambda_i} + \frac{\alpha_{1,2}\alpha_{2,1}(\mu_2^{MSR\text{-}1} - \mu_1^{MSR\text{-}1})^2}{\lambda_i(\mu^* - \lambda_i)(\alpha_{1,2} + \alpha_{2,1})^3} + B,$$

$$\textit{where average server speed } \mu^* = \frac{\alpha_{1,2}\mu_2^{MSR\text{-}1} + \alpha_{2,1}\mu_1^{MSR\text{-}1}}{\alpha_{1,2} + \alpha_{2,1}},$$

$$B \geq \frac{\alpha_{2,1}(\mu_1^{MSR\text{-}1} - \mu_2^{MSR\text{-}1})}{\lambda_i(\alpha_{1,2} + \alpha_{2,1})^2},$$

$$\textit{and } B \leq \frac{\alpha_{1,2}(\mu_2^{MSR\text{-}1} - \mu_1^{MSR\text{-}1})}{\lambda_i(\alpha_{1,2} + \alpha_{2,1})^2} + \frac{\max_j \left\{\mathbf{u}_i^j\right\}}{\lambda_i}.$$

Proof. This follows from Theorem 2 and the bound on MSR-1 systems derived in [4]. □

Figure 1 shows these bounds compared to simulation.

The Optimized MSR Policy. Although many MSR policies can be used to stabilize a given system, our response time bounds in Theorem 2 tell us which MSR policy one should use to minimize mean response time. Specifically, we can compute an *Optimized MSR Policy* (OMP) that minimizes our response time upper bound. To simplify this discussion, we consider the concrete example of finding an OMP in the two-state example described in Corollary 1.

First, observe that when $\alpha_{1,2} + \alpha_{2,1} \to \infty$, $E[T_i^\pi] \to 1/(\mu^* - \lambda_i)$. That is, our optimized policy can preempt running jobs infinitely frequently with no overhead, it is optimal to let the transition rates of the modulating process go to infinity. Then, we can pick our MSR policy by minimizing the lead term in our response time bound, which corresponds to the system load in the MSR-1 system.

In practice, jobs cannot be preempted infinitely frequently. Hence, we also consider finding an OMP subject to a *preemption budget*, $\bar{\alpha}$, that limits the long-run average rate of preemption the scheduling policy can perform. In this case, we choose an OMP via the following minimization problem:

$$\min_{\pi, S^\pi} \sum_i \lambda_i \cdot (\text{upper bound on } \mathbb{E}[T_i^\pi])$$

$$\text{s.t. } \alpha \leq \bar{\alpha}, \text{ and } \mathbb{E}[\mathbf{u}_\pi] \cdot \boldsymbol{\mu} > \boldsymbol{\lambda}.$$

Here, α is the average preemption rate for a given choice of MSR policy. In general, α is straightforward to calculate from stationary distribution of the modulating process, $\{\pi(t)\}$. Given our two-state example, for instance, we have

$$\alpha = \frac{2\alpha_{1,2}\alpha_{2,1}}{\alpha_{1,2} + \alpha_{2,1}}.$$

Note that as $\bar{\alpha} \to 0$, the $(\mu_2^{MSR\text{-}1} - \mu_1^{MSR\text{-}1})^2$ term in our bound blows up. Here, the OMP will choose the schedules that minimize $(\mu_2^{MSR\text{-}1} - \mu_1^{MSR\text{-}1})^2$.

Our response time bounds depend on three factors: (i) the average system load, (ii) the rate at which the modulating process changes schedules, and (iii) the variability in the schedules the policy uses. If an MSR policy changes schedules frequently, the variability factor vanishes, since jobs roughly experience a system with a constant completion rate, the average server speed μ^*. If the policy preempts infrequently, the variability factor will dominate. Given a fixed preemption budget, $\bar{\alpha}$, the OMP must balance a tradeoff between system load and the variability of its schedules.

4. CONCLUSION

We consider the problem of scheduling multiresource jobs. We prove response time bounds for a simple, throughput-optimal class of scheduling policies called MSR policies. We derive bounds on the mean response time of any MSR policy, and show how these bounds can be used to choose the best MSR policy for a given system.

5. REFERENCES

[1] Eryilmaz, A., and Srikant, R. Asymptotically tight steady-state queue length bounds implied by drift conditions. *Queueing Systems 72* (2012), 311–359.

[2] Fabila-Monroy, R., and Huemer, C. Caratheodory's theorem in depth. *Discrete & Computational Geometry 58* (2017), 51–66.

[3] Ghaderi, J. Randomized algorithms for scheduling vms in the cloud. In *IEEE INFOCOM* (2016), IEEE.

[4] Grosof, I., Hong, Y., Harchol-Balter, M., and Scheller-Wolf, A. The RESET and MARC techniques, with application to multiserver-job analysis. *Performance Evaluation 162* (2023), 102378.

[5] Kleinrock, L. *Queueing systems, volume 2: Computer applications*, vol. 66. Wiley New York, 1976.

[6] Maguluri, S. T., Srikant, R., and Ying, L. Heavy traffic optimal resource allocation algorithms for cloud computing clusters. *Performance Evaluation* (2014).

[7] Psychas, K., and Ghaderi, J. Randomized algorithms for scheduling multi-resource jobs in the cloud. *IEEE/ACM ToN 26*, 5 (2018), 2202–2215.