heSRPT:Parallel Scheduling to Minimize Mean Slowdown

Benjamin Berg*
Carnegie Mellon University
bsberg@cs.cmu.edu

Rein Vesilo
Macquarie University
rein.vesilo@mq.edu.au

Mor Harchol-Balter†
Carnegie Mellon University
harchol@cs.cmu.edu

Introduction

Modern data centers serve workloads which can exploit parallelism. When a job parallelizes across multiple servers it completes more quickly. However, it is unclear how to share a limited number of servers between many parallelizable jobs.

In this paper we consider a typical scenario where a data center composed of N servers will be tasked with completing a set of M parallelizable jobs. Typically, M is much smaller than N. In our scenario, each job consists of some amount of inherent work which we refer to as a job's *size*. We assume that job sizes are known up front to the system, and each job can utilize any number of servers at any moment in time. These assumptions are reasonable for many parallelizable workloads such as training neural networks using TensorFlow [2]. Our goal in this paper is to allocate servers to jobs so as to minimize the *mean slowdown* across all jobs, where the slowdown of a job is the job's completion time divided by its running time if given exclusive access to all N servers. Slowdown measures how a job was interfered with by other jobs in the system, and is often the metric of interest in the theoretical parallel scheduling literature (where it is also called stretch), as well as the HPC community (where it is called expansion factor).

What makes this problem difficult is that jobs receive a concave, sublinear speedup from parallelization – jobs have a decreasing marginal benefit from being allocated additional servers (see Figure 1). Hence, in choosing a job to receive each additional server, one must keep the overall efficiency of the system in mind. This paper aims to determine the optimal allocation of servers to jobs where jobs follow a realistic sublinear speedup function.

It is clear that the optimal allocation policy will depend heavily on the jobs' speedup – how parallelizable the jobs being run are. To see this, first consider the case where each job is *embarrassingly parallel* (see Figure 1), and can be parallelized perfectly across an arbitrary number of servers. In this case, we observe that the entire data center can be viewed as a *single server* that can be perfectly utilized by or shared between jobs. Hence, from the single server scheduling literature, it is known that the Shortest Remaining Processing Time policy (SRPT) will minimize the mean slowdown across jobs [3]. By contrast, if we consider the case where jobs are hardly parallelizable, a single job receives very little benefit from additional servers. In this case, the optimal policy is to divide the system equally between jobs, a policy called EQUI. In practice, a realistic speedup function usually lies somewhere between these two extremes and thus we must balance a trade-off between the SRPT and EQUI policies in order to minimize mean slowdown. Specifically, since jobs are *partially* parallelizable, it is still beneficial to allocate more servers to smaller jobs than to large jobs. The optimal policy with respect to mean slowdown must split the difference between these policies, favoring short jobs while still respecting the overall efficiency of the system.

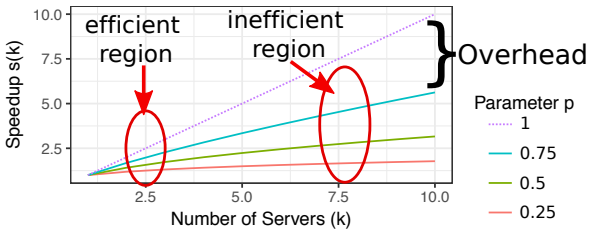

Figure 1: A variety of speedup functions of the form $s(k) = k^p$, shown with varying values of p. When $p = 1$ we say that jobs are *embarrassingly parallel*, and hence we consider cases where $0 < p < 1$. All functions in this family are concave and lie below the embarrassingly parallel speedup function ($p = 1$).

Overview of Results

Our model assumes there are N identical servers which must be allocated to M parallelizable jobs. All M jobs are present at time $t = 0$. Job i is assumed to have some inherent size x_i where, without loss of generality,

$$x_1 \geq x_2 \geq \ldots \geq x_M.$$

We assume that all jobs follow the *same* speedup function, $s : \mathbb{R}^+ \to \mathbb{R}^+$, which is of the form

$$s(k) = k^p$$

for some $0 < p < 1$. Specifically, if a job i of size x_i is allocated k servers, it will complete at time

$$\frac{x_i}{s(k)}.$$

*This author is supported by a Facebook Graduate Fellowship
†This author was supported by: NSF-CMMI-1938909, NSF-CSR-1763701, NSF-XPS-1629444, and a Google 2020 Faculty Research Award.

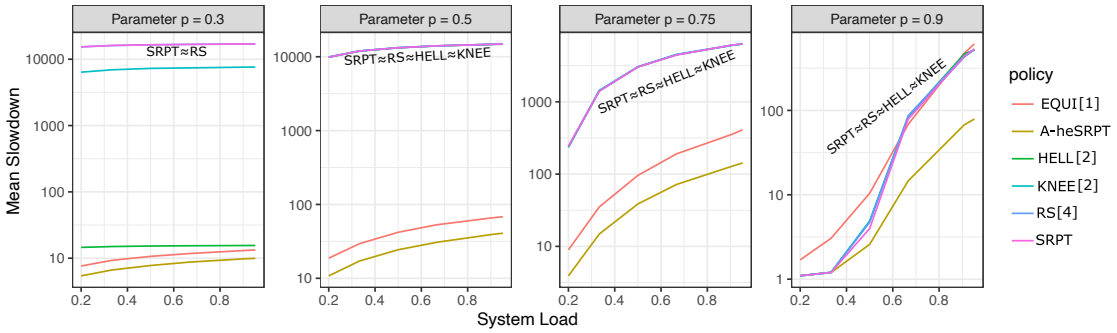

Figure 2: Mean slowdown of A-heSRPT and policies from the literature in the online setting. Simulations assume $N = 10,000$ servers and a Poisson arrival process. Job sizes are Pareto($\alpha = 1.5$) distributed. Each graph shows mean slowdown for one value of the speedup parameter, p, where $s(k) = k^p$. A-heSRPT often dominates by an order of magnitude.

In general, the number of servers allocated to a job can change over the course of the job's lifetime. It therefore helps to think of $s(k)$ as a *rate* of service where the remaining size of job i after running on k servers for a length of time t is

$$x_i - t \cdot s(k).$$

An *allocation policy*, P, must determine how many servers are allocated to each job at every moment in time. We denote the completion time of job i under policy P as T_i^P. Hence, the slowdown of job i under policy P, S_i^P, can be written as

$$S_i^P = \frac{T_i^P}{x_i/s(N)}.$$

Our goal is to derive an allocation policy that minimizes the *mean slowdown* across jobs. That is, we wish to find the optimal function

$$\boldsymbol{\theta}^*(t) = (\theta_1^*(t), \theta_2^*(t), \ldots, \theta_M^*),$$

where $\theta_i^*(t)$ denotes the fraction of the N servers allocated to job i at any moment in time t.

To derive the optimal allocation policy, we develop a new technique to reduce the dimensionality of the optimization problem. This dimensionality reduction leverages two key properties of the optimal policy. First, we show that the optimal policy must complete jobs in shortest-job-first order. This claim is stated in Theorem 1.

THEOREM 1 (OPTIMAL COMPLETION ORDER). *The optimal policy completes jobs in Shortest-Job-First order:*

$$M, M-1, M-2, \ldots, 1.$$

Next, we prove the scale-free property, which illustrates the optimal substructure of the optimal policy. Our scale-free property states that for any job, i, job i's allocation relative to jobs completed after job i (jobs larger than job i) is constant throughout job i's lifetime. This property is stated formally in Theorem 2.

THEOREM 2 (SCALE-FREE PROPERTY). *Under the optimal policy, for any job, i, which completes at time T_i^*, there exists a constant c_i^* such that, for all $t < T_i^*$*

$$\frac{\theta_i^*(t)}{\sum_{j=1}^{i} \theta_j^*(t)} = c_i^*.$$

Theorems 1 and 2 allow us to reduce our optimization problem to a single, unconstrained optimization problem of M variables. We solve this simplified optimization problem to derive the first closed form expression for the optimal allocation of servers to jobs which minimizes the mean slowdown of a set of M jobs.

The optimal allocation function is given by Theorem 3.

THEOREM 3 (OPTIMAL ALLOCATION FUNCTION). *At time t, when $m(t)$ jobs remain in the system,*

$$\theta_i^*(t) = \begin{cases} \left(\frac{z(i)}{z(m(t))}\right)^{\frac{1}{1-p}} - \left(\frac{z(i-1)}{z(m(t))}\right)^{\frac{1}{1-p}} & 1 \le i \le m(t) \\ 0 & i > m(t) \end{cases}$$

where

$$z(k) = \sum_{i=1}^{k} \frac{1}{x_i/s(N)}.$$

Our optimal allocation balances the benefits of SRPT and the high efficiency of EQUI. We thus refer to our optimal policy as *high efficiency SRPT* (heSRPT). We also provide a closed form expression for the slowdown under heSRPT. Additionally, we show that heSRPT can be generalized to minimize a class of weighted flow time metrics that includes both mean slowdown and mean flow time.

While most of this paper deals with the setting where all jobs are present at time 0, the online setting where jobs arrive over time is equally important. We propose an online version of heSRPT called *Adaptive-heSRPT* which uses the allocations from heSRPT to recalculate server allocations on every arrival and departure. We compare A-heSRPT to several policies from the literature — HELL[2], KNEE[2], EQUI[1], and RS[4]. Adaptive-heSRPT outperforms these competitor policies in simulation, often by an order of magnitude. These simulation results are shown in Figure 2.

1. REFERENCES

[1] B. Berg, J.P. Dorsman, and M. Harchol-Balter. Towards optimality in parallel scheduling. *ACM POMACS*, 1(2), 2018.

[2] S. Lin, M. Paolieri, C. Chou, and L. Golubchik. A model-based approach to streamlining distributed training for asynchronous SGD. In *MASCOTS*. IEEE, 2018.

[3] Donald R Smith. A new proof of the optimality of the shortest remaining processing time discipline. *Operations Research*, 26(1):197–199, 1978.

[4] Adam Wierman, Mor Harchol-Balter, and Takayuki Osogami. Nearly insensitive bounds on SMART scheduling. *SIGMETRICS*, 33(1):205–216, 2005.