Network Speed Scaling

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

Speed scaling for a network of servers represented by a directed acyclic graph is considered. Jobs arrive at a source server, with a specified destination server, and are defined to be complete once they are processed by all servers on any feasible path between the source and the corresponding destination. Each server has variable speed, with power consumption function P, a convex increasing function of the speed. The objective is to minimize the sum of the flow time (summed across jobs) and the energy consumed by all the servers, which depends on how jobs are routed, as well as how server speeds are set. Algorithms are derived for both the worst case and stochastic job arrivals setting, whose competitive ratio depends only on the power functions and path diversity in the network, but is independent of the workload.

1. INTRODUCTION

Starting with the classical work [11], the speed scaling problem has been widely studied in the literature, where there is a single server or a parallel bank of servers with tuneable speed, and the canonical problem is to find the optimal service speed/rate for servers that minimizes a linear combination of the flow time (total delay) and total energy [1], called flow time plus energy, where flow time is defined as the sum of the response times (departure minus the arrival time) across all jobs.

Many interconnected practical systems such as assembly lines, flow shops and job shops in manufacturing, traffic flow in a network of highways, multihop telecommunications networks, and client-server computer systems, however, are better modelled as network of queues/servers. Another important example is service systems with server specific precedence constraints, where jobs have to be processed by a particular order of servers. In such systems, for each job, service is defined to be complete once it has been processed by a subset of servers, together with a permissible order on service from different servers. However, in spite of the extensive literature on speed scaling for a parallel bank of servers (surveyed below), we are not aware of any work that addresses speed scaling on a service *network*.

This paper takes the first steps towards designing speed scaling algorithms for a network of servers. Jobs corresponding to different source destination pairs arrive into the network, and are considered complete once they are served sequentially by all servers on a feasible (directed) path from the source to the destination. There are three decision variables here: (i) *routing*, i.e., choosing the service path that each job takes, (ii) *scheduling*, i.e., choosing which waiting job each server processes, and (iii) *speed scaling*, i.e., deciding the speed with which each server processes its jobs.

We consider the online scenario, where the algorithm has only causal information about job arrivals that arrive according to either an arbitrary or a stochastic process. In the arbitrary (also called worst case) job arrival setting, the job arrival sequence is arbitrary, and possibly adversarially determined. In the arbitrary job arrivals case, the goal is to minimize the algorithm cost, defined as a linear combination of the flow time (the sum total of the response times of all jobs) and the total energy consumption, and the performance metric is the competitive ratio, that is defined as the maximum of the ratio of the cost of the online algorithm and the optimal offline algorithm OPT that is revealed the entire input sequence ahead of time, over all possible inputs. In the stochastic setting, job arrivals occur according to a stochastic process. Here, the cost of an algorithm is a linear combination of the steady state averages of response time and energy consumption per job. The competitive ratio of an algorithm is in turn the ratio of its cost to that of the optimal algorithm (that admits the above steady state averages). In both settings, the goal is to design algorithms that have a small competitive ratio.

2. MAIN RESULTS

The contributions of this paper are summarized below.

Arbitrary input: In the worst case setting, jobs corresponding to each source destination pair (a.k.a., flow) arrive at the source at arbitrary times, the arrival times being possibly chosen by an adversary. For the most part, we make the simplifying assumption that each job has the same (unit, without loss of generality) size (a.k.a, service requirement) on any server. Indeed, even in the single server setting, initial progress was made assuming unit sized jobs (see [1, 8, 4, 6, 2]), which was later generalized for arbitrary job sizes [3, 5]. In the network setting, this problem is even harder, with additional complications arising from from precedence constraints between servers, dynamic routing, as well as intra-flow and inter-flow congestion. Throughout, we also assume that all servers have the same power function $P(.)$ (though this is generalized in the stochastic setting).

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

We consider three specific network models in this paper, in increasing order of generality.

Tandem network with single source destination pair: This is the simplest network setting, with multiple servers (K) in series. Each external job arrives at server 1, and is defined to be *complete* once it has been processed by each of the K servers in series. Let $n^1(t)$ be the number of outstanding jobs with server 1, and let the total number of servers with outstanding jobs (called active) be $A(t)$ excluding the first server. Then the algorithm runs each active server (including server 1) at the same speed of $P^{-1}\left(\frac{n^1(t)+A(t)+1}{A(t)+1}\right)$. Thus, the total power consumed across all servers is equal to the number of total outstanding jobs plus 1. We show that this algorithm is constant competitive, with a competitive ratio that depends only on the power function. Crucially, the competitive ratio does not depend on the size of the tandem network.

We also show that the preceding analysis can be extended to the case where job sizes are arbitrary in the *burst* setting, i.e., all jobs are available at time 0.

Single source destination pair, multiple parallel paths: Next, we consider the setting where is single source destination pair with multiple parallel paths. When each parallel path has the same length, we propose an algorithm that has constant competitive ratio similar to the tandem setting. If the available parallel paths have different lengths, our competitive ratio depends on the path diversity.

Multiple source destination pairs, multiple parallel paths: Finally, we consider the general case where the service network is shared by different flows. Note that in this model, different flows corresponding to distinct source-destination pairs can 'intersect' at a server. Our competitive ratio in this case is a function of the maximum degree across servers, where the degree of a server is the number of active paths belonging to distinct source-destination pairs going through that server. Importantly, the competitive ratio is independent of the size of the network, the number of flows (source-destination pairs) or the number of jobs for each flow.

Stochastic input: In the stochastic setting, we consider a more general service network, where each flow is associated with a set of (possibly intersecting) feasible paths. Each server j has its own power function $P_j(.)$. For most part we consider, $P_j(s) = \gamma_j s^{\alpha_j}$. The external arrivals corresponding to each flow are assumed to follow a Poisson process, and service times at each server being expoentially distributed. Our algorithm routes each flow based on the solution of a certain convex program, and employs a simple 'gated' static speed. This makes the queueing system a feedforward Jackson network with Poisson arrivals at each server [7]. We show that the proposed algorithm has a constant competitive ratio of $\theta_1\theta_2$, where θ_1 only depends on the power functions $P_j's$, while θ_2 depends on the path diversity in the network.

The factor θ_1 in the competitve ratio bound depends only on the server power functions $P_j's$, where $P_j(s) = \gamma_j s^{\alpha_j}$. If $\alpha_j = \alpha$ and $\gamma_j = 1$ for all servers j, then $\theta_1 = 1 + 2\gamma_j + \max(1, 2\alpha_j - 2)$, which evaluates to 5 for $\alpha = 2$ and 7 for $\alpha = 3$. This bound can be further tightened for smaller values of α by modifying the speed scaling rule. For example, the alternative speed scaling rule proposed in [9] would result in $\theta_1 = 2$ (without affecting θ_2) for $\alpha = 2$.

The factor θ_2 depends on the network topology, the path diversity for different flows, and the server costs (which depend on the power functions). In particular,

$$\theta_2 = \max_{i \in \text{flows}} \theta_{2i},$$

where for flow i, $\theta_{2i} = \sum_{p_i \in \text{paths for flow } i} \lambda^\star_{p_i} \frac{L_{p_i}}{L_i^{\min}}$, where $\lambda^\star_i$ is the fraction of jobs routed along path p_i, where routing is based on the solution

If there is only one permissible path for each flow, then $\theta_2 = 1$. More generally, for each flow f, if the ratio of the length of any path for f and the length of the shortest path for f is upper bounded by a constant c, then $\theta_2 \le c$. Thus, for a balanced network, θ_2 is bounded. If the network is not bounded, i.e., the length of some path p is 'much longer' than the shortest path for a flow, then the routing algorithm will allocate only a small fraction of traffic to p, and then again θ_2 will be small.

For all theorem statements and proofs, we refer to the full version [10].

3. REFERENCES

[1] S. Albers and H. Fujiwara. Energy-efficient algorithms for flow time minimization. *ACM Transactions on Algorithms (TALG)*, 3(4):49, 2007.

[2] S. Albers, F. Müller, and S. Schmelzer. Speed scaling on parallel processors. *Algorithmica*, 68(2):404–425, 2014.

[3] L. L. H. Andrew, M. Lin, and A. Wierman. Optimality, fairness, and robustness in speed scaling designs. In *Proc. ACM SIGMETRICS*, 2010.

[4] N. Bansal, H.-L. Chan, T.-W. Lam, and L.-K. Lee. Scheduling for speed bounded processors. In *International Colloquium on Automata, Languages, and Programming*, pages 409–420. Springer, 2008.

[5] N. Bansal, H.-L. Chan, and K. Pruhs. Speed scaling with an arbitrary power function. In *ACM-SIAM symposium on discrete algorithms*, 2009.

[6] N. Bansal, K. Pruhs, and C. Stein. Speed scaling for weighted flow time. *SIAM Journal on Computing*, 39(4):1294–1308, 2009.

[7] M. Harchol-Balter. *Performance modeling and design of computer systems: queueing theory in action.* Cambridge University Press, 2013.

[8] T.-W. Lam, L.-K. Lee, I. K. To, and P. W. Wong. Speed scaling functions for flow time scheduling based on active job count. In *European Symposium on Algorithms*, pages 647–659. Springer, 2008.

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

[10] R. Vaze and J. Nair. Network speed scaling. *Performance Evaluation*, page 102145, 2020.

[11] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced cpu energy. In *Annual Symposium on Foundations of Computer Science*, 1995.