https://pubsonline.informs.org/journal/moor

MATHEMATICS OF OPERATIONS RESEARCH
Articles in Advance, pp. 1–33
ISSN 0364-765X (print), ISSN 1526-5471 (online)

A New Approach to Capacity Scaling Augmented with Unreliable Machine Learning Predictions

Daan Rutten,[a] **Debankur Mukherjee**[a,*]

[a] H. Milton Stewart School of Industrial & Systems Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332
*Corresponding author
Contact: drutten@gatech.edu, https://orcid.org/0000-0002-4742-4201 (DR); debankur.mukherjee@isye.gatech.edu, https://orcid.org/0000-0003-1678-4893 (DM)

Received: February 13, 2021
Revised: November 4, 2021; October 26, 2022
Accepted: February 27, 2023
Published Online in Articles in Advance: April 13, 2023

MSC2020 Subject Classifications: Primary: 68T05, 68W27; secondary: 68M20

https://doi.org/10.1287/moor.2023.1364

Abstract. Modern data centers suffer from immense power consumption. As a result, data center operators have heavily invested in capacity-scaling solutions, which dynamically deactivate servers if the demand is low and activate them again when the workload increases. We analyze a continuous-time model for capacity scaling, where the goal is to minimize the weighted sum of flow time, switching cost, and power consumption in an online fashion. We propose a novel algorithm, called adaptive balanced capacity scaling (ABCS), that has access to black-box machine learning predictions. ABCS aims to adapt to the predictions and is also robust against unpredictable surges in the workload. In particular, we prove that ABCS is $(1+\varepsilon)$ competitive if the predictions are accurate, and yet, it has a uniformly bounded competitive ratio even if the predictions are completely inaccurate. Finally, we investigate the performance of this algorithm on a real-world data set and carry out extensive numerical experiments, which positively support the theoretical results.

Funding: This work was partially supported by the Division of Computing and Communication Foundations [Grant 2113027]. The authors also acknowledge financial support for this project from the Algorithm and Randomness Center–Transdisciplinary Research Institute for Advancing Data Science Fellowship at Georgia Tech.

Keywords: energy efficiency • online algorithms • competitive analysis • speed scaling • competitive ratio

1. Introduction

1.1. Background and Motivation

Modern data centers suffer from immense power consumption, which amounts to a massive economic and environmental impact. In 2014, data centers alone contributed to about 1.8% of the total U.S. electricity consumption (Shehabi et al. [53]), and this is projected to reach 7% in 2030 (Jones [29]). Consequently, data center providers are constantly striving to optimize their servers for energy efficiency, pushing the hardware's efficiency to nearly its limit. At this point, algorithmic improvements appear to be critical in order to achieve substantial further gain (Shehabi et al. [53]). A common practice for data centers has been to reserve significant excess service capacity in the form of idle servers (Sverdlik [54]), even though a typical idle server still consumes about 44% of its peak power consumption (Shehabi et al. [53]). The recommendation from the U.S. Department of Energy (Shehabi et al. [53]), industry (Facebook [18], Google [26], Netflix [48]), and the academic research community (Albers and Fujiwara [1], Gandhi et al. [22], Lin et al. [36]) is, therefore, to implement dynamic capacity-scaling functionality based on the demand. If the demand is low, the service capacity should be scaled down by deactivating servers, whereas at peak times, it should be scaled up by increasing the number of active servers. Instead of physically toggling servers on or off, this functionality is often implemented by carefully allocating a fraction of servers to other lower-priority services and quickly bringing them back at times of high demand; see Cortez et al. [14], Rzadca et al. [52], and Tirmazi et al. [55] for a more detailed account. This maximizes the utilization of the system and hence, minimizes the power consumption.

The call for algorithmic solutions to capacity scaling has inspired a vibrant line of research over the last decade (Albers and Fujiwara [1], Augustine et al. [6], Bansal et al. [9], Gandhi et al. [20], Gandhi et al. [22], Irani et al. [28], Lin et al. [36], Lu et al. [37], Mukherjee and Stolyar [46], Mukherjee et al. [47]). The problem fits into the framework of online algorithms, where the goal is to design algorithms that dynamically scale the current service capacity based on the past and current system information. Here, the performance of an algorithm is captured in terms of the competitive ratio (CR), which is defined as the worst possible ratio between the cost incurred by the online algorithm and that by the offline optimum algorithm. Note that the online algorithm has information only about the past and the present, whereas the offline optimum has accurate information about all future input variables, such as the task-arrival process

in the context of the current article. The key advantage of such strong performance guarantees lies in its robustness; that is, the algorithm safeguards against the worst-case scenario.

However, any of today's modern large-scale systems has access to massive historical data, which combined with standard machine learning (ML) algorithms, can reveal definitive patterns. In these cases, simply following the recommendations obtained from the ML predictions typically outperforms any competitive algorithm. Netflix is an example of a company implementing capacity scaling in practice. Instead of relying on competitive online algorithms, Netflix has implemented ML algorithms in their Scryer system (Netflix [48]). They noted that their demand usually follows regular patterns, allowing them to accurately predict the demand during a day based on data from previous weeks. Most of the time, the performance of the machine learning algorithm is, therefore, excellent. However, besides empirical verification, the performance of such ML predictions is not guaranteed. In fact, repeated observations show that unexpected surges in the workload are not at all uncommon (Bodik [12], Lassettre et al. [34], Netflix [48]), and they cause a significant adverse impact on the system performance.

The contrasting approaches between academia and industry reveal a gap between what we are able to prove and what is desirable in practice. Although online algorithms do not require any information about future arrivals, in practice, these predictions are usually available. At the same time, an algorithm should not blindly trust the predictions because occasionally, the accuracy of the predictions can be significantly poor. The current work aims to bridge this gap by incorporating ML predictions directly into the competitive analysis framework. In particular, we propose a novel low-complexity algorithm for capacity scaling called adaptive balanced capacity scaling (ABCS), which has access to a black-box predictor, lending predictions about future arrivals. Critically, not only is ABCS completely unaware of the prediction's accuracy, we also restrain from making *any* statistical assumptions on the accuracy. Hence, this excludes any attempt to learn the prediction's accuracy because accurate past predictions do not necessarily warrant the quality of future predictions. The main challenge, therefore, is to design near-optimal algorithms that intelligently accept and reject the recommendations given by the ML predictor without knowing or learning their accuracy. Note, however, that the performance of ABCS does depend on the (unknown) error of the prediction, and it ensures, among others, two desirable properties: (i) *consistency* (i.e., if the predictions turn out to be accurate in hindsight, then ABCS automatically nearly replicates the optimal solution) and (ii) *competitiveness* (i.e., if the predictions are inaccurate in hindsight, then the performance of ABCS is at most a uniformly bounded constant factor times the minimum cost). The formal definitions of consistency and competitiveness are given in Section 3. It is worth emphasizing that this work is not concerned with how the ML predictions are obtained and uses them as a black box.

1.2. Our Contributions

We will use a canonical continuous-time dynamical system model that is used to analyze algorithms for energy efficiency; see, for example, Albers and Fujiwara [1], Bansal et al. [9], Lin et al. [36], Lu et al. [37], and Maccio and Down [39] for variations. Consider a system with a large number of homogeneous servers. Each server is in either of two states: *active* or *inactive*. Let $m(t)$ denote the number of active servers at time t. Workload arrives into the system in continuous time and gets processed at instantaneous rate $m(t)$. The system has a buffer of infinite capacity, where the unprocessed workload can wait until it is executed. We will assume that there is an unknown and arbitrary arrival rate function $\lambda(t)$ that represents the arrival process; see Section 2 for further details. We do not impose any restrictions on $\lambda(\cdot)$. To contrast this with the often-studied case when the workload arrival is stochastic, $\lambda(\cdot)$ can be thought of as an individual sample path of the corresponding stochastic arrival process. At any time, the system may decide to increase or decrease $m(t)$ in an online fashion. However, it pays a switching cost each time a server is activated. This represents the cost of terminating the lower-priority service running at the inactive server and related migration costs (Lin et al. [36], Lu et al. [37], Maccio and Down [39], Rzadca et al. [52]). The goal of the system is to minimize the weighted sum of the flow time, the switching cost, and the power consumption (Maccio and Down [39]). The flow time is defined as the total time tasks spend in the system and is a measure of the response time (Albers and Fujiwara [1], Bansal et al. [9]). We will analyze the performance of an algorithm by its competitive ratio, the worst-case ratio between the cost of the online algorithm and the minimum offline cost, over all possible arrival rate functions $\lambda(\cdot)$. We further assume that the algorithm receives predictions about future workload through an ML oracle (Lykouris and Vassilvtiskii [38], Mahdian et al. [40]). More precisely, at time $t = 0$, the ML oracle predicts an arrival rate function $\tilde{\lambda}(\cdot)$. The algorithm may use these predictions to increase or decrease the number of servers accordingly. For instance, if the oracle predicts that the demand in the next hour will increase, then the algorithm might proactively increase the number of servers. However, as mentioned before, it is crucial that the algorithm is completely oblivious to the accuracy of these predictions. We measure the accuracy of the predictions in terms of the mean absolute error (MAE) between the predicted arrival rate function $\tilde{\lambda}$ and the actual rate function λ (see Definition 1). Our contributions in the current paper are threefold.

1. Purely online algorithm with worst-case guarantees. First, we propose a novel purely online algorithm for capacity scaling called balanced capacity scaling (BCS). This purely online scenario, or the scenario of traditional competitive analysis, is equivalent to having predictions with infinite error. There are several fundamental works that have considered the purely online scenario for capacity scaling (Ghandi et al. [20], Lin et al. [36], Lu et al. [37], Mukherjee and Stolyar [46], Mukherjee et al. [47]). We extend the state of the art in this area by analyzing a general model in continuous time where unprocessed workload is allowed to wait. In fact, we show that a class of popular algorithms that were previously proposed is not constant competitive in this more general case (see Proposition 4). We show that BCS is five competitive in the general case (Corollary 1) and is two competitive when waiting is not allowed and workload must be processed immediately upon arrival (Theorem 2). BCS is easy to implement and is memoryless (i.e., it only depends on the current state of the system and not on the past). Also, we prove a lower-bound result that any deterministic online algorithm must have a competitive ratio of at least 2.549 (Proposition 2), which implies that the problem is strictly harder than the classical ski rental problem, a benchmark for online algorithms.

2. Augmenting unreliable ML predictions. When ML predictions are available, we first propose an adaptive algorithm called adapt to the prediction (AP), which ensures consistency. That is, we prove (Theorem 3) that the competitive ratio of AP is at most $1 + \Theta(\eta)$, where η is a suitable measure of the prediction's accuracy and is a function of the MAE between the predicted arrival rate function $\tilde{\lambda}$ and the actual rate function λ (Definition 2). AP does not follow the predictions blindly. Rather, it dynamically scales the number of servers in an online fashion as the past predictions turn out to be inaccurate. Although the performance of AP is optimal as $\eta = 0$ and it degrades gracefully with η, it is not constant competitive if predictions are completely inaccurate ($\eta = \infty$). Thus, it does not provide any worst-case guarantees. This is a feat shared by many recent adaptive algorithms in the literature (see Remark 6).

Next, we combine the ideas behind BCS and AP to propose an algorithm that is both competitive *and* consistent. This brings us to the main contribution of the paper. We propose ABCS, which uses the structure of BCS and utilizes AP as a subroutine. ABCS has a hyperparameter $r \geq 1$, which can be fixed at any value before implementing the algorithm, and represents our confidence in the ML predictions. If we choose $r = 1$, then the algorithm works as a purely online one and disregards all predictions. In this case, ABCS is five competitive, as before. However, for any fixed $r > 1$, we prove (Corollary 2) that the competitive ratio of ABCS is at most

$$\mathrm{CR}(\eta) \leq \min((1 + \mathcal{O}(\eta)) \cdot (1 + r^{-1} + \mathcal{O}(r^{-2})), \mathcal{O}(r^{7/2})), \tag{1.1}$$

where η is the prediction's accuracy as before. There are a number of consequences of the result. We start by emphasizing that although the competitive ratio is a function of the error η, the algorithm is completely oblivious to it. Now, the higher we fix the value of r to be, the closer the competitive ratio of ABCS gets to one if the predictions turn out to be accurate in hindsight. If the predictions are completely inaccurate ($\eta = \infty$), the competitive ratio is at most $\mathcal{O}(r^{7/2})$, a constant that depends only on r and *not* on η. ABCS is, therefore, robust against unpredictable surges in workload while providing near-optimal performance if the predictions are accurate.

Another interesting thing to note is that for $r > 1$, the competitive ratio in (1.1) is the minimum of two terms; the first term, which we call the *optimistic competitive ratio* (OCR), is smaller when the prediction is accurate, and the second term, which we call the *pessimistic competitive ratio* (PCR), is smaller when the prediction is inaccurate. From the algorithm designer's perspective, there is a clear trade-off between OCR and PCR, which is conveniently controlled by the confidence hyperparameter r. It is important to note that ABCS provides performance guarantees for *any fixed* $r \geq 1$ irrespective of the model parameters or the accuracy of the predictions. However, the choice of r reflects the risk that the system designer is willing to take in the pessimistic case against the gain in the optimistic case. See Remark 10 for further discussion. This trade-off, however, is not specific to our algorithm. In fact, we prove a negative result in Proposition 3 that *any* algorithm that is $(1 + \delta)$ competitive in the optimistic case has a competitive ratio of at least $1/(4\delta)$ in the pessimistic case.

3. Offline algorithm for regular workloads. Finally, we consider the scenario in which the workload $\lambda(\cdot)$ is known perfectly up front. We propose an offline algorithm that solves a linear program and prove (Theorem 5) that if the workload is sufficiently regular (see Assumption 1), then the offline algorithm is $(1 + \mathcal{O}(\delta))$ competitive with respect to the offline optimal algorithm. Here, δ is a hyperparameter of the algorithm that measures the desired accuracy. As δ decreases, the accuracy of the solution increases; however, the dimension of the linear program (in terms of the number of decision variables and constraints) increases at rate $1/\delta$ as well. The offline algorithm may be used as a subroutine in AP, even if the predictions are unreliable.

To test the performance of our algorithms in practice, we implemented them on both a real-world data set of domain name system (DNS) requests observed at a campus network (Manmeet et al. [41]) and a set of artificial data sets, and the performance turned out to be excellent. See Section 5 for details.

1.3. Related Works

Over the past two decades, the rapid growth of data centers and their immense power consumption have inspired a vibrant line of research in optimizing the energy efficiency of such systems (Barroso and Hölzle [11], Dayarathna et al. [16], Rong et al. [51], Urgaonkar et al. [56]). We provide an overview of a few influential works relevant to the current paper.

The capacity-scaling problem was introduced in a seminal paper by Lin et al. [36], who analyze a discrete-time model of a data center. At each time step t, the cost of operating $m(t)$ servers is determined by the switching cost and an arbitrary convex function $g_t(m(t))$, which for example, specifies the cost of increased power consumption versus response time. At time step t, the system reveals the function g_t and accurate functions $g_{t+1}, g_{t+2}, \dots, g_{t+w}$ in a prediction window of w future time steps. Lin et al. [36] propose an algorithm, called the lazy capacity provisioning (LCP) algorithm, and prove that it is three competitive. Surprisingly, the performance of the LCP algorithm does not improve if $w > 0$ (i.e., if predictions are available). We consider a modified model in continuous time, where predictions are not necessarily accurate. Moreover, the performance of our algorithm increases provably in the presence of predictions.

Lu et al. [37] consider a scenario where tasks cannot wait in queue and must be served immediately upon arrival. They discover that in this case, the capacity-scaling problem reduces to solving a number of independent ski rental problems. The authors then propose an algorithm and prove that it is two competitive. Our model, in addition, includes the response time, which directly generalizes the framework of Lu et al. [37]. This flexibility introduces a whole new dimension in the space of possible decisions. For example, because the results of Lu et al. [37] lack any form of delay, tasks are processed at the same time by any algorithm. Our model allows an algorithm-dependent delay of serving tasks, which *desynchronizes* the time at which tasks are processed at a server across different algorithms and hence, significantly complicates the analysis. Mazzucco and Dyachuk [42] analyze a related problem, in which the number of servers is periodically updated and a task is lost if a server is not immediately available to serve it. The goal of their algorithm is to balance the power consumption and the cost of losing tasks. Galloway et al. [19] and later, Gandhi et al. [21] and Gandhi et al. [23] perform empirical studies of data centers. Their results show that significant power savings are possible while maintaining much of the latency of the network.

A well-studied problem that is somewhat related to our setup is *speed scaling*. Here, the goal is to optimize the processing speed of a single server and to minimize the weighted sum of the flow time and power consumption, whereas the switching cost is zero. The power consumption is typically cubic in the processing speed. In contrast to our model, the scheduling of jobs also plays a crucial role here. A seminal paper in this area is by Bansal et al. [9], who propose an algorithm that schedules the task with the shortest remaining processing time first and processes it at a speed such that the power consumption is equal to the number of waiting tasks plus one. The authors prove that this algorithm is $(3 + \varepsilon)$ competitive. Later papers have extended the case of the single server to processor-sharing systems (Wierman et al. [57]) and parallel processors with deadline constraints (Albers et al. [2]). The problem of speed scaling has also been analyzed in the case that the interarrival times and required processing times are exponentially distributed (Ata and Shneorson [5]).

Any algorithm for the capacity-scaling problem consists of two components: first, to activate servers and second, to deactivate servers. For a single server, a natural abstraction of the latter problem is the famous ski rental problem, as first introduced by Karlin et al. [31]. The ski rental problem has been applied to cases of capital investment (Azar et al. [7], Damaschke [15]), TCP acknowledgement (Karlin et al. [30]), and cache coherence (Anderson and Karlin [3]). Irani et al. [28] analyze the ski rental problem when multiple power-down states are available, such as active, sleeping, hibernating, and inactive. The power consumption in each state is different, and moving between the states incurs a switching cost. Augustine et al. [6] generalize these results when the transition costs between the different states are not additive. Although the current work focuses on only two states (i.e., active and inactive), we expect that the algorithm and proofs are general enough to accommodate multiple power-down states, which we leave as interesting future work. Khanafer et al. [32] analyze the ski rental problem in a stochastic context.

Two papers are often independently credited for initiating the study of online algorithms augmented by ML predictions: Lykouris and Vassilvtiskii [38] in the context of caching and Mahdian et al. [40] in the context of allocation of online advertisement space, load balancing, and facility location. Lykouris and Vassilvtiskii [38] show how to adapt the marker algorithm for the caching problem to obtain a competitive ratio of two if the predictions are perfectly accurate and a bounded competitive ratio if the predictions are inaccurate. Mahdian et al. [40] propose an algorithm that

naively switches between an optimistic scheduling algorithm and a pessimistic scheduling algorithm to minimize the make span when routing tasks to multiple machines. We here mostly follow the terminology of Lykouris and Vassilvtiskii [38]. Since then, the ideas have been applied to bipartite matching (Kumar et al. [33]), ski rental and scheduling on a single machine (Purohit et al. [49]), bloom filters (Mitzenmacher [43]), and frequency estimation (Hsu et al. [27]). Lee et al. [35] propose an algorithm that operates on-site generators to reduce the peak energy usage of data centers. Although related to the current work, their algorithm works independent to the capacity scaling happening inside the data center. Bamas et al. [8] discuss an algorithm augmented by predictions for the related problem of speed scaling discussed, in the case of parallel processors with deadline constraints. Similar to our results, Bamas et al. [8] identify a trade-off between what we call an optimistic competitive ratio and a pessimistic competitive ratio. We prove, for the capacity-scaling problem considered in the current work, that *any* algorithm must exhibit such a trade-off (see Proposition 3 for details). Antoniadis et al. [4] discuss algorithms augmented with predictions for the general framework of metrical task systems. Note that the problem in the current paper cannot be described in the form of a metrical task system. For example, if one casts the problem in a metric space that contains both the number of servers and the workload in the queue, then the possible transitions depend on a nontrivial combination of the arrival function, the number of servers, and the workload in the queue. The metrical task system allows the possible transitions to depend on either the metric between two points or the arrival function but not a combination thereof. However, one cannot omit the number of servers or the workload in the queue from the metric space either because the cost depends on both. Therefore, the workload in the queue adds a completely new way in which decisions between rounds are coupled that cannot be captured by a metric space.

Recently, the notion of a predictor has also emerged in stochastic scheduling. Mitzenmacher [44] introduces the predictor as a probability density function $g(x, y)$ for a task with actual service time x and predicted service time y. Here, the author analyzes the shortest predicted job first and shortest predicted remaining processing time queueing disciplines for a single queue and determines the price of misprediction (i.e., the ratio of the cost if perfect information of the service time distribution is known and the cost if only predictions are available). For multiple queues, Mitzenmacher [45] has simulated the supermarket model or the "power-of-d" model to show empirically that the availability of predictions greatly improves performance.

A different line of work called online algorithms with advice questions how many bits of *perfect* future information are necessary to reproduce the optimal offline algorithm (see Boyar et al. [13] for a survey). The difference with the current work is that we do not assume that the predictions are perfect but instead, have arbitrary accuracy.

When the arrival process and service times are stochastic, there are several major works that consider energy efficiency of the system. Gandhi et al. [20] provide an exact analysis of the M/M/k/setup system. The system is similar to the M/M/k class of Markov chains (i.e., tasks arrive according to a Poisson process and require an exponentially distributed processing time). To process the tasks, the system has access to a maximum of k servers. According to the algorithm in Gandhi et al. [20], if a task arrives and there are no available servers, the system moves one server to its setup state, where it remains for an exponentially random time before the server becomes active. The authors provide a sophisticated method to analyze the system exactly. Maccio and Down [39] analyze a similar system for a broader class of cost functions. When each server has a dedicated separate queue, Mukherjee and Stolyar [46] and Mukherjee et al. [47] analyze the case where the setup times and standby times (the time a server remains idle before it is deactivated) are independent exponentially distributed. In this case, they propose an algorithm that achieves asymptotic optimality for both the response time and the power consumption in the large-system limit. Earlier research has also modeled the response time as a constraint rather than charging a cost for the response time (Goldman et al. [25]). Here, each task is presented with a deadline; the task should be served before this deadline, or it is irrevocably lost. The earliest deadline first queueing discipline has been proven to be effective in this case (Doytchinov et al. [17]).

1.4. Notation and Organization

The remainder of the paper is organized as follows. Section 2 describes the model. Section 3 introduces some preliminary concepts and definitions related to the ML predictions, such as the error. Section 4 introduces our algorithms and the main results, of which the high-level proof ideas are provided in Section 6. Most of the technical proofs are given in the appendix. Section 5 presents extensive numerical experiments, including the performance of our algorithms on a real-world data set. Finally, Section 7 concludes our work and presents directions for future research.

2. Model Description

We now introduce a general model for capacity scaling. Let ω, β, and θ be fixed nonnegative parameters of the model. We will assume that the tasks waiting in the buffer accumulate a waiting cost at rate $\omega > 0$, the cost of activating a

server is $\beta > 0$, and each active server accumulates a power consumption cost at rate $\theta \geq 0$. We denote the workload in the buffer at time t by $q(t)$.

An instance consists of a known finite-time horizon $T > 0$ and an unknown and arbitrary function $\lambda : [0,T] \to \mathbb{R}_+$ representing the arrival process. The model is

$$\begin{aligned} \underset{m:[0,T]\to\mathbb{R}_+}{\text{minimize}} \quad & \omega \cdot \int_0^T q(s)\mathrm{d}s + \beta \cdot \limsup_{\delta \downarrow 0} \sum_{i=0}^{\lfloor T/\delta \rfloor} [m(i\delta+\delta) - m(i\delta)]^+ + \theta \cdot \int_0^T m(s)\mathrm{d}s \\ \text{subject to} \quad & q(t) = \int_0^t (\lambda(s) - m(s))\mathbb{1}\{q(s) > 0 \text{ or } \lambda(s) \geq m(s)\}\mathrm{d}s \text{ for all } t \in [0,T] \\ & m(0) = 0,\ m(t) \geq 0 \text{ for all } t \in (0,T], \end{aligned} \tag{2.1}$$

where $[x]^+ = \max(x,0)$. To solve the optimization problem, an algorithm needs to determine the function $m(\cdot)$ given the parameters ω, β, θ. Note that our goal is to investigate *online* algorithms, meaning that $\lambda(\cdot)$ is revealed to the algorithm in an online fashion. In other words, at time t, the algorithm must determine $m(t)$ depending only on $\lambda(s)$ for $s \in [0,t]$. Note that the system may equivalently reveal the total workload received before time $s \in [0,t]$, as λ is simply its rate of increase. For an algorithm that runs $m(t)$ servers at time t, the cost accumulated until time t is defined as

$$\textsc{Cost}^\lambda(m,t) := \omega \cdot \int_0^t q(s)\mathrm{d}s + \beta \cdot \limsup_{\delta \downarrow 0} \sum_{i=0}^{\lfloor t/\delta \rfloor} [m(i\delta+\delta) - m(i\delta)]^+ + \theta \cdot \int_0^t m(s)\mathrm{d}s. \tag{2.2}$$

We will compare the total cost $\textsc{Cost}^\lambda(m,T)$ for an online algorithm with that of the *offline* minimum defined as

$$\textsc{Opt} := \inf_{m:(0,T]\to\mathbb{R}_+} \textsc{Cost}^\lambda(m,T), \tag{2.3}$$

and without loss of generality, we will assume $\textsc{Opt} < \infty$ throughout the paper.

Remark 1. The minimizer of (2.3) exists, as stated by the next proposition. The proof of Proposition 1 is given in Appendix A.1. The difficulty in the proof is in dealing with the second term in (2.2), which makes the function $\textsc{Cost}^\lambda(m,T)$ discontinuous in $m(\cdot)$ with respect to the L_1 norm.

Proposition 1. *There exists* $m^* : [0,T] \to \mathbb{R}_+$ *such that* $\textsc{Cost}^\lambda(m^*,T) = \textsc{Opt}$.

Remark 2. The model in (2.1) assumes that $m(0) = 0$ for the sake of clarity of exposition. The results in this paper extend to any $m(0)$ by adding an additive constant of $\mathcal{O}(\beta \cdot m(0))$ to each of the performance bounds. The proofs in the appendix are presented for this more general case.

The model in (2.1) actually combines some well-studied state-of-the-art models (Albers and Fujiwara [1], Bansal et al. [9], Lin et al. [36], Lu et al. [37], Maccio and Down [39]). To see how it relates to the problem of capacity scaling, note that the objective function in (2.1) is a weighted sum of three metrics. We clarify each of them. These three metrics are common performance measures of the system, such as the response time or the power consumption. The parameters ω, β, and θ represent the weights assigned to each of these metrics. The three metrics are as follows.

i. The flow time. The flow time is defined as the total time a task spends in the system and captures the response time of the system. Note that the average response time per unit of workload is $\frac{\int_0^T q(s)\mathrm{d}s}{\int_0^T \lambda(s)\mathrm{d}s}$; see also Albers and Fujiwara [1], and Bansal et al. [9]. The weight ω is the cost attributed to the response time (e.g., in dollars per second). The weight ω could, for example, be determined based on loss of revenue or user dissatisfaction as a result of increased response time.

ii. The switching cost. As in Lin et al. [36], Lu et al. [37], and Rzadca et al. [52], the parameter β can be viewed as the cost to increase the number of active servers (e.g., in dollars per server). This may include, for example, the cost to terminate a lower-priority service and related migration costs. In practice, these costs are usually equivalent to the cost of running the server for multiple hours (Lin et al. [36]). The total switching cost is β times the number of times a server is made active.

iii. The power consumption. The power consumption is proportional to the total time servers are in the active state (Lu et al. [37]). The weight θ represents the cost of power (e.g., in dollars per server per second).

Also, the constraints in (2.1) model the dynamics of capacity scaling, and $q(\cdot)$ can be viewed as the queue-length process or the remaining workload process. Note that (2.1) does not require $q(t)$ or $m(t)$ to be integer valued. This is a fairly standard relaxation because a service may typically request a fraction of the server's capacity

(Rzadca et al. [52], Tirmazi et al. [55]) and a single task is tiny; see, for example, Lin et al. [36] and Mukherjee et al. [47]. The system in (2.1) can also be interpreted as a *fluid counterpart* of a discrete system. Figure 1 depicts the model schematically.

Remark 3. The model in (2.1) assumes that the service capacity can be increased nearly instantaneously. Hence, it does not include the so-called setup time. Besides being a common assumption in competitive analysis (see, for example, Lin et al. [36] and Lu et al. [37]), this is also not completely unreasonable in practice. This is mainly because servers are not usually physically turned off in reality. Instead, when a server becomes "inactive," the server's capacity will be used by other low-priority services. Then, activating a server means quickly terminating such low-priority services; see Rzadca et al. [52] and Tirmazi et al. [55] for a more detailed account. From a theoretical standpoint, the assumption of a zero setup time is also necessary to get a uniformly bounded competitive ratio, as stated in the next lemma. For the sake of Lemma 1, let us assume that in the capacity-scaling problem in (2.1), there is an additional setup time $t_0 > 0$ before the number of servers can be increased. In other words, if the online algorithm decides to turn on a server at time t, then the number of servers is increased at time $t + t_0$. The proof of Lemma 1 is provided in Appendix A.2.

Lemma 1. *Let $\mathcal{A}$ be any deterministic algorithm for the capacity-scaling problem in* (2.1)*, and assume that there is an additional setup time $t_0 > 0$ before the number of servers can be increased. Also, let* CR *denote the competitive ratio of $\mathcal{A}$ (see Definition 3 for a formal definition of the competitive ratio). Then, there exists θ such that* $\mathrm{CR} \geq \frac{\omega t_0^2}{2\beta}$*. In short, there does not exist any deterministic online algorithm with a uniformly bounded competitive ratio.*

Formulation (2.1) is fairly easy to solve as an *offline* optimization problem. Section 4.3 presents a linear program that solves the offline problem. However, as mentioned earlier, we are interested in an *online* algorithm. Specifically, we distinguish two scenarios.

1. Purely online scenario. The system reveals ω, β, θ, and at time t, also $\lambda(s)$ for $s \in [0, t]$ to the online algorithm but not $\lambda(s)$ for any $s > t$. The purely online scenario corresponds to the setting where predictions may not be available and provides a natural starting point of our investigation. We discuss a competitive algorithm for the purely online scenario in Section 4.1. Additionally, in this purely online scenario, our algorithm does not require the system to reveal the finite-time horizon T up front.

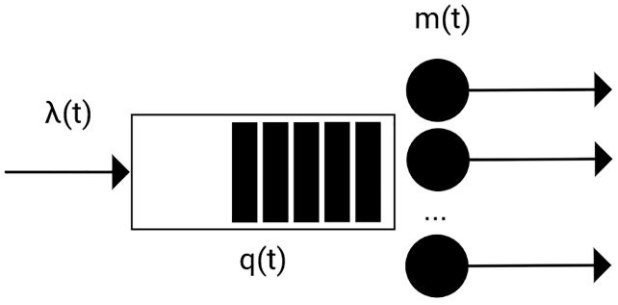
2. Machine learning scenario. In addition to the assumptions in the purely online scenario, at time $t = 0$, an ML predictor predicts the arrival rate function of the entire interval; that is, it predicts the arrival rate function to be $\tilde{\lambda} : [0, T] \to \mathbb{R}_+$. The ML predictor may, for example, be trained on the past observed workload on a day. For the purpose of the current work, we treat the predictor as a black box. We discuss a consistent algorithm for the machine learning scenario in Section 4.2.1, for which the competitive ratio degrades gracefully with the prediction's accuracy. However, the algorithm is not competitive in the worst case. Finally, in Section 4.2.2, we discuss an algorithm for the machine learning scenario, which is simultaneously competitive and consistent, by combining the algorithms from Sections 4.1 and 4.2.1.

The idea of using online algorithms with unreliable machine-learned advice was first introduced in Mahdian et al. [40] in the context of allocation of online advertisement space, load balancing, and facility location and in Lykouris and Vassilvtiskii [38] in the context of competitive caching. The next section provides the necessary details of the framework of Lykouris and Vassilvtiskii [38].

3. Preliminary Concepts

This section briefly presents the competitive analysis framework for algorithms that have access to ML predictions. We mostly follow the setup as introduced in Lykouris and Vassilvtiskii [38] and adapt it here for the current scenario.

Figure 1. The system receives tasks at rate $\lambda(t)$ and operates $m(t)$ servers. The workload is $q(t)$.



We measure the errors in predictions by the MAE between the true and predicted label, which is commonly used in state-of-the-art machine learning algorithms (Gao [24], Qi et al. [50]).

Definition 1. The error in the prediction $\tilde{\lambda}(\cdot)$ with respect to the actual arrival rate $\lambda(\cdot)$ is

$$\|\tilde{\lambda} - \lambda\|_{MAE} = \frac{1}{T}\int_0^T |\tilde{\lambda}(t) - \lambda(t)|\,\mathrm{d}t. \tag{3.1}$$

To measure the performance of an algorithm augmented by an ML predictor, we will define the competitive ratio as a function of the prediction's accuracy. However, before stating the definition of competitive ratio, we introduce the level of accuracy of a prediction.

Definition 2. Fix a finite-time horizon T and arrival rate function $\lambda(\cdot)$. Let Opt be as defined in (2.3). For $\eta > 0$, we say that a prediction $\tilde{\lambda}$ is η accurate for the instance (T, λ) if

$$\|\tilde{\lambda} - \lambda\|_{MAE} \le \eta \cdot \frac{\mathrm{Opt}}{T}. \tag{3.2}$$

The definition of the prediction's accuracy is intimately tied to the cost of the optimal solution. As already argued in Lykouris and Vassilvtiskii [38], because Opt is a linear functional of λ, normalizing the error by the cost of the optimal solution is necessary. This is because the definition should be invariant to scaling and padding arguments. For example, if we double both $\lambda(\cdot)$ and $\tilde{\lambda}(\cdot)$, then the prediction's accuracy should still be the same.

Let $\mathcal{A}$ be any algorithm for (2.1). The performance of $\mathcal{A}$ is measured by the competitive ratio $\mathrm{CR}(\eta)$, which itself is a function of the accuracy η. The following definition is an adaptation of Lykouris and Vassilvtiskii [38, definition 3] for the current setup.

Definition 3. Fix a finite-time horizon T and arrival rate function $\lambda(\cdot)$. Let $\mathcal{A}$ be any algorithm for (2.1), and let $m(t)$ denote its number of servers when it has access to a prediction $\tilde{\lambda}$ and Opt be as defined in (2.3). We say that $\mathcal{A}$ has a competitive ratio at most CR for the instance (T, λ) and prediction $\tilde{\lambda}$ if

$$\mathrm{Cost}^{\lambda}(m, T) \le \mathrm{CR} \cdot \mathrm{Opt}. \tag{3.3}$$

We say that the competitive ratio of $\mathcal{A}$ is at most $\mathrm{CR}(\eta)$ if the competitive ratio is at most $\mathrm{CR}(\eta)$ for all instances (T, λ) and any η-accurate prediction $\tilde{\lambda}$. By convention, we say that $\mathrm{CR}(\eta) = \infty$ if such a finite $\mathrm{CR}(\eta)$ does not exist.

Note that although the competitive ratio depends on the prediction's accuracy, the algorithm is oblivious to this accuracy. We desire three properties of an online algorithm that has access to a prediction. The algorithm's performance should (i) be close to the optimal solution if the prediction is perfect, (ii) degrade gracefully with the prediction's error, and (iii) be bounded regardless of the prediction's accuracy. The definitions of consistency, robustness, and competitiveness summarize these desiderata; see also Lykouris and Vassilvtiskii [38, definitions 4–6].

Definition 4. Let $\mathcal{A}$ be any algorithm for (2.1) and $\mathrm{CR}(\eta)$ denote its competitive ratio when it has access to an η-accurate prediction. Then, we say that

i. Algorithm $\mathcal{A}$ is ρ consistent if $\mathrm{CR}(0) = \rho$,
ii. Algorithm $\mathcal{A}$ is α robust if $\mathrm{CR}(\eta) = \mathcal{O}(\alpha(\eta))$, and
iii. Algorithm $\mathcal{A}$ is γ competitive if $\mathrm{CR}(\eta) \le \gamma$ for all $\eta \in [0, \infty]$.

4. Main Results

4.1. Balanced Capacity-Scaling Algorithm

We first discuss a competitive algorithm in the purely online scenario. Recall that in this case, the system reveals ω, β, θ, and at time t, also $\lambda(s)$ for $s \in [0, t]$ to the algorithm but not $\lambda(s)$ for any $s > t$. Moreover, as mentioned earlier, the results in this section also hold when the finite-time horizon T is not revealed up front. The BCS algorithm that we propose is parameterized by two nonnegative numbers r_1 and r_2. Algorithm 1 describes BCS for any fixed choices of r_1 and r_2.

Algorithm 1 (BCS (r_1, r_2))

Choose $m(\cdot)$ such that at each time $t \ge 0$,

$$\frac{\mathrm{d}m(t)}{\mathrm{d}t} = \frac{r_1\omega \cdot q(t) - r_2\theta \cdot m(t)}{\beta}. \tag{4.1}$$

We start by briefly discussing the intuition behind BCS. At each time $t \geq 0$, BCS computes the derivative of the number of servers (i.e., how fast the system should increase or decrease the service capacity). Note that if we solve Equation (4.1), then we obtain the number of servers $m(t)$, which is differentiable for all $t \geq 0$. The two parameters r_1 and r_2 control how fast the algorithm reacts by increasing or decreasing the number of servers, respectively. If the workload $q(t)$ is nonzero, then the first term in the right-hand side of Equation (4.1) increases the number of servers at rate r_1. The second term is an "inertia term," which decreases the number of servers at rate r_2. Note that if we integrate Equation (4.1), we obtain

$$\int_0^t r_1 \omega \cdot q(s) \mathrm{d}s = \int_0^t \beta \cdot \frac{\mathrm{d}m(s)}{\mathrm{d}s} \mathrm{d}s + \int_0^t r_2 \theta \cdot m(s) \mathrm{d}s. \tag{4.2}$$

This means that BCS aims to carefully balance the flow time with the switching cost plus the power consumption. BCS is memoryless and computationally cheap. The derivative of the number of servers depends only on the current workload and number of servers, without requiring knowledge about the past workload, number of servers, or arrival rate. BCS can, therefore, be implemented without any memory requirements.

We are able to characterize the performance of BCS analytically for any fixed choices of r_1 and r_2. Theorem 1 characterizes the competitive ratio of BCS. The proof of Theorem 1 is provided in Section 6.1.

Theorem 1. *Let CR denote the competitive ratio of BCS (Algorithm 1). Then,*

$$\mathrm{CR} \leq \left(1 + \frac{1}{r_1} + \frac{1}{r_2}\right) \max(2, r_1, 2r_2). \tag{4.3}$$

The optimal choice of the parameters is $r_1 = 2$ and $r_2 = 1$. Corollary 1 states that BCS is five competitive in this case.

Corollary 1. *Let CR denote the competitive ratio of BCS (Algorithm 1). If $r_1 = 2$ and $r_2 = 1$, then* $\mathrm{CR} \leq 5$.

Moreover, in the special case when tasks are not allowed to wait and must be served immediately upon arrival ($\omega = \infty$), BCS turns out to be two competitive, as stated in Theorem 2. Note that the algorithm introduced by Lu et al. [37] in the special case $\omega = \infty$ is also two competitive and that the authors prove that this is, in fact, optimal. The proof of Theorem 2 is given in Appendix A.4.

Theorem 2. *Let CR denote the competitive ratio of BCS (Algorithm 1). If $r_1 = 2$, $r_2 = 1$, and $\omega = \infty$, then* $\mathrm{CR} \leq 2$.

Note that the capacity-scaling problem has previously been related to the classical ski rental problem (Augustine et al. [6], Irani et al. [28], Lu et al. [37]), which is two competitive. As it turns out, when tasks are allowed to wait, the formulation in (2.1) of the capacity-scaling problem is strictly harder than the ski rental problem, as Proposition 2 states. Proposition 2 is proved in Appendix A.5.

Proposition 2. *Let $\mathcal{A}$ be any deterministic algorithm for the capacity-scaling problem in (2.1) in the purely online scenario, and* CR *denotes its competitive ratio. There exist choices for ω, β, and θ such that* $\mathrm{CR} \geq 2.549$. *In other words, any deterministic algorithm is at least* 2.549 *competitive.*

Remark 4. We should note that the proof of Proposition 2 assumes that the finite-time horizon T is not revealed up front. We leave it to future work to identify a (possibly weaker) lower bound if T is known to the algorithm.

4.2. Augmenting Unreliable ML Predictions

To augment BCS with machine learning predictions, we proceed in two steps. First, in Section 4.2.1, we introduce AP. We prove that the competitive ratio of AP degrades gracefully with the prediction's accuracy, although AP is not competitive. Second, in Section 4.2.2, we discuss how to combine BCS and AP to obtain ABCS, which follows the predictions but is robust against inaccurate predictions and therefore, competitive.

4.2.1. Adapt to the Prediction Algorithm. We will now turn our attention to the machine learning scenario. Recall that in this case, at time $t = 0$, the algorithm receives a predicted arrival rate function $\tilde{\lambda} : [0, T] \to \mathbb{R}_+$. Note that a trivial way to implement the predictions is to blindly trust the predictions: that is, to let

$$m \in \operatorname*{arg\,min}_{m:(0,T] \to \mathbb{R}_+} \textsc{Cost}^{\tilde{\lambda}}(m, T). \tag{4.4}$$

The minimum exists (see Remark 1). However, in this case, the performance decays drastically even for relatively small prediction errors. Indeed, if the actual arrival rate $\lambda(\cdot)$ is higher than the predicted arrival rate $\tilde{\lambda}(\cdot)$ at the start, then the associated workload could stay in the queue until the end of the time horizon $[0, T]$ and incur a significant waiting

cost. We instead propose AP, which consists of an offline component and an online component. The offline component computes an estimate for the number of servers up front based on $\tilde{\lambda}(\cdot)$. The online component follows the offline estimate but dynamically adapts the number of servers based on discrepancies between the predicted and actual arrival rates. Let us define

$$\Delta\lambda(t) := \begin{cases} \left(\lambda(t) - \tilde{\lambda}(t)\right)^+ & \text{for} \quad t \geq 0 \\ 0 & \text{for} \quad t < 0. \end{cases}$$

Algorithm 2 describes AP.

Algorithm 2 (AP)

Choose $m(\cdot)$ such that at each time $t \geq 0$,

$$m(t) = m_1(t) + m_2(t), \tag{4.5}$$

where

$$m_1 \in \underset{m:(0,T]\to\mathbb{R}_+}{\arg\min} \ \textsc{Cost}^{\tilde{\lambda}}(m, T), \tag{4.6}$$

$$\frac{\mathrm{d}m_2(t)}{\mathrm{d}t} = \sqrt{\frac{\omega}{2\beta}} \cdot \left(\Delta\lambda(t) - \Delta\lambda\left(t - \sqrt{2\beta/\omega}\right)\right). \tag{4.7}$$

The number of servers under AP consists of two components, an offline component m_1 and an online component m_2. The offline component m_1 is determined up front by the optimal number of servers to handle the predicted arrival rate $\tilde{\lambda}$. The online component m_2 is determined in an online manner, and it reacts if the actual arrival rate turns out to be higher than the predicted arrival rate. Note that if we solve Equation (4.7), then we obtain the number of servers $m_2(t)$, which is differentiable for all $t \geq 0$. The online component works as follows. If $\Delta\lambda(t) > 0$, then the online component increases the service capacity at rate $\sqrt{\omega/(2\beta)}$. In other words, for each additional unit of workload received, the number of servers is increased by $\sqrt{\omega/(2\beta)}$. After a fixed time of $\sqrt{2\beta/\omega}$, the number of servers is decreased again. Intuitively, if $\omega \gg \beta$, then the online component turns on many servers for a short period of time, whereas if $\beta \gg \omega$, then the online component turns on a few servers for a longer period of time.

Remark 5. The constants $\sqrt{\omega/(2\beta)}$ and $\sqrt{2\beta/\omega}$ in (4.7) are chosen to minimize the competitive ratio of AP. More specifically, one can prove a similar performance guarantee as in Theorem 3 but for any arbitrary choice of these constants. The choice of $\sqrt{\omega/(2\beta)}$ and $\sqrt{2\beta/\omega}$ is the unique minimizer of the competitive ratio. Hence, AP in its current form outperforms any other choice of constants in the worst case.

Although the optimization in the offline component might be expensive, it has to be performed only once at the start. To solve the minimization problem, one could, for example, use the offline approximation technique, which we describe in Section 4.3. Moreover, if the predictions are based on historical data, the offline component m_1 might even be precomputed and retrieved from memory at the start. The online component, in contrast, is computationally cheap.

The competitive ratio of AP, of course, depends on the accuracy of the predictions. Theorem 3 characterizes the performance of AP. Recall the definition of the competitive ratio $\mathrm{CR}(\eta)$ from Section 3.

Theorem 3. *Fix any finite-time horizon T, arrival rate function $\lambda(\cdot)$, and prediction $\tilde{\lambda}(\cdot)$. Let $m(t)$ be the number of servers of AP (Algorithm 2) and* Opt *be as defined in (2.3). Then,*

$$\textsc{Cost}^{\lambda}(m, T) \leq \textsc{Opt} + (\sqrt{2\omega\beta} + \theta)T \cdot \|\tilde{\lambda} - \lambda\|_{MAE}. \tag{4.8}$$

Let $\mathrm{CR}(\eta)$ denote the competitive ratio of AP (Algorithm 2) when it has access to an η-accurate prediction. Then, as a result of this,

$$\mathrm{CR}(\eta) \leq 1 + (\sqrt{2\omega\beta} + \theta)\eta. \tag{4.9}$$

The proof of Theorem 3 is provided in Appendix A.6. If η is small, then the competitive ratio is close to one. In fact, AP replicates the optimal solution exactly if the predictions turn out to be accurate and hence, is one consistent. Moreover, the competitive ratio also degrades gracefully in the prediction's accuracy, which as discussed earlier, is not achieved by the offline component m_1 alone.

Remark 6. Although AP does not follow the predictions blindly, AP is not competitive because it is not hard to verify that $\mathrm{CR}(\eta) \to \infty$ as $\eta \to \infty$ (e.g., let $\tilde{\lambda}(t) \to \infty$ uniformly for all $t \in [0, T]$). Note that earlier algorithms proposed in the literature, such as the receding horizon control and LCP algorithms from Lin et al. [36], are proven to be competitive only if predictions are accurate (i.e., consistent in the terminology of the current paper). As these algorithms follow the predictions blindly, these algorithms are, therefore, not competitive if predictions are inaccurate. Hence, the goal in the next subsection is to combine the approaches of BCS and AP to obtain an algorithm that follows the predictions most of the time but ignores the predictions when appropriate.

Remark 7. The competitive ratio bound is scale invariant in the weights ω, β, and θ. For example, if each of the weights ω, β, and θ is doubled, then the factor $\sqrt{2\omega\beta} + \theta$ doubles as well. However, because Opt doubles, the accuracy η is halved (recall Definition 2).

4.2.2. Adaptive Balanced Capacity Scaling.

We now answer the question of how to follow the predictions most of the time without trusting them blindly. The ABCS algorithm we propose strategically combines BCS and AP introduced earlier. Let $\tilde{m}(\cdot)$ be the number of servers as turned on by AP (Algorithm 2). Let $\tilde{q}(\cdot)$ be the queue-length process under AP: that is,

$$\tilde{q}(t) = \int_0^t (\lambda(s) - \tilde{m}(s)) \mathbb{1}\{\tilde{q}(s) > 0 \text{ or } \lambda(s) \geq \tilde{m}(s)\} \mathrm{d}s \qquad \text{for all } t \geq 0. \tag{4.10}$$

ABCS is parameterized by four nonnegative numbers $R_1 \geq r_1 \geq 0$ and $R_2 \geq r_2 \geq 0$. Algorithm 3 describes ABCS for any fixed choices of R_1, r_1, R_2, r_2.

Algorithm 3 (ABCS (r_1, r_2, R_1, R_2))

Choose $m(\cdot)$ such that at each time $t \geq 0$,

$$\frac{\mathrm{d}m(t)}{\mathrm{d}t} = \frac{\hat{r}_1(t)\omega \cdot q(t) - \hat{r}_2(t)\theta \cdot m(t)}{\beta}, \tag{4.11}$$

where

$$\hat{r}_1(t) = \begin{cases} r_1 & \text{if } m(t) - \tilde{m}(t) > [q(t) - \tilde{q}(t)]^+ \cdot \sqrt{\dfrac{\omega}{2\beta}}, \\ R_1 & \text{if } m(t) - \tilde{m}(t) \leq [q(t) - \tilde{q}(t)]^+ \cdot \sqrt{\dfrac{\omega}{2\beta}}, \end{cases}$$
$$\hat{r}_2(t) = \begin{cases} R_2 & \text{if } m(t) > \tilde{m}(t) \text{ and } q(t) \leq \tilde{q}(t), \\ r_2 & \text{if } m(t) \leq \tilde{m}(t) \text{ or } q(t) > \tilde{q}(t). \end{cases} \tag{4.12}$$

Remark 8. It is worthwhile to highlight that ABCS is oblivious to the choice of AP as the source of the advised number of servers $\tilde{m}(\cdot)$. Therefore, if there exists an algorithm similar to AP but with a better error dependence, then it is straightforward to extend ABCS to use this algorithm as the source for the advised number of servers instead. The improved error dependence carries over immediately into the competitive ratio of ABCS (see also Proposition 6).

In spirit, ABCS works similarly to BCS. In fact, if $R_1 = r_1$ and $R_2 = r_2$, then ABCS is equivalent to BCS and disregards predictions altogether. However, in contrast to the constant rates r_1 and r_2 of BCS, the rates at which ABCS reacts are captured by the state-dependent rate functions $\hat{r}_1(t)$ and $\hat{r}_2(t)$. The reason behind the precise choices of $\hat{r}_1(t)$ and $\hat{r}_2(t)$ will be clear later from the performance of the algorithm. From a high-level perspective, these are chosen to approach the behavior of the advised number of servers $\tilde{m}(t)$ of AP. Indeed, if ABCS has less than the advised number of servers $\tilde{m}(t)$, then it increases $m(t)$ at the higher rate R_1 and decreases it at the lower rate r_2. Similarly, if ABCS has "sufficiently more" servers than the advised number $\tilde{m}(t)$, then it increases $m(t)$ at the lower rate r_1 and decreases it at the higher rate R_2. The number of servers of ABCS,, therefore judiciously approaches the number of advised servers. However, it does not blindly follow $\tilde{m}(t)$ to protect against inaccurate predictions. For example, if the workload $q(t)$ is significantly higher than the current number of servers $m(t)$, then ABCS will always increase the number of servers at a nonzero rate.

Our main result characterizes the performance of ABCS analytically, which is presented in Theorem 4. The proof of Theorem 4 is provided in Section 6.3. Recall the definition of the competitive ratio $\mathrm{CR}(\eta)$ from Section 3.

Theorem 4. *Let* $\mathrm{CR}(\eta)$ *denote the competitive ratio of ABCS (Algorithm 3) when it has access to an* η*-accurate prediction. Then, for any* $R_1 \geq r_1 \geq 0$ *and* $R_2 \geq r_2 \geq 0$,

$$\mathrm{CR}(\eta) \leq \min\Big((1+(\sqrt{2\omega\beta}+\theta)\eta)\cdot \mathrm{OCR}, \mathrm{PCR}\Big), \tag{4.13}$$

where

$$\begin{aligned}
&\mathrm{OCR} = \max\left(c_1 r_1, \frac{c_2 R_1}{\sqrt{1+2R_1}}, c_2 + c_3, c_4\right), \quad \mathrm{PCR} = \max\left(c_5 R_1, 2c_6, 2c_6 R_2 + 1 - \frac{R_2}{r_2}\right),\\
&c_1 = 1 + \frac{1}{r_1} + \frac{1}{R_2}, \quad c_2 = \frac{c_1\sqrt{1+2r_1} - c_1 + c_3}{\sqrt{1+2R_1}}, \quad c_3 = 1 + \frac{1}{R_1} + \frac{1}{R_2},\\
&c_4 = 1 + r_2 + \frac{r_2}{R_1} + c_2 r_2, \quad c_5 = 1 + \frac{1}{r_1} + \frac{1}{r_2}, \quad c_6 = c_5\sqrt{\frac{R_1}{r_1}}.
\end{aligned} \tag{4.14}$$

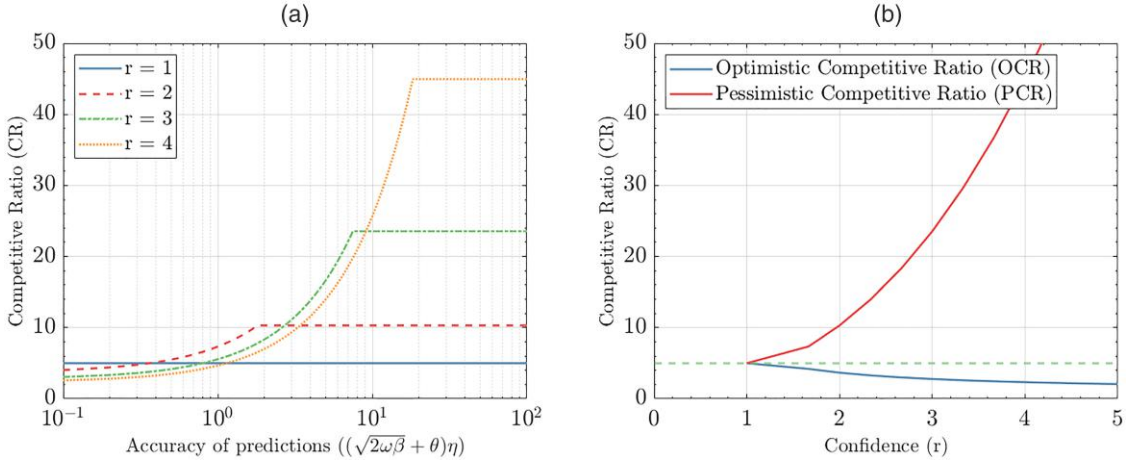
Theorem 4 characterizes the competitive ratio of ABCS explicitly for any choices of the parameters. Note that for any value of η, the competitive ratio is at most PCR. Moreover, if η is small, then the competitive ratio is close to OCR. It is straightforward to check from Theorem 4 that ABCS satisfies the three desiderata of Definition 4. In particular, ABCS is OCR consistent and PCR competitive. The constants OCR and PCR, of course, depend on the parameters R_1, r_1, R_2, and r_2. Corollary 2 provides guidance on how to choose these parameters asymptotically optimally.

Corollary 2. *Let* $r \geq 1$ *be a hyperparameter, representing the confidence in the predictions. Let* $\mathrm{CR}(\eta)$ *denote the competitive ratio of ABCS (Algorithm 3) when it has access to an* η*-accurate prediction. If* $R_1 = 8(r-1)^2 + 2$, $r_1 = (r-0.5)^{-1}$, $R_2 = 2r-1$, *and* $r_2 = r^{-1}$, *then*

$$\begin{aligned}
\mathrm{CR}(\eta) \leq \min\Big((1+(\sqrt{2\omega\beta}+\theta)\eta)\cdot\left(1+\frac{1}{r}+\frac{9}{8r^2}+\mathcal{O}\left(\frac{1}{r^3}\right)\right),\\
(2r-1)(4r+1)\sqrt{(2r-1)(4r(r-2)+5)}\Big).
\end{aligned} \tag{4.15}$$

Corollary 2 characterizes the trade-off between the OCR and the PCR. If the confidence in the predictions r is set at a high value, then the OCR tends to one. However, the value of PCR tends to become large in this case, even though importantly, it remains uniformly bounded as $\eta \to \infty$. Figure 2(a) plots the competitive ratio as a function of η and the confidence hyperparameter r. For fixed r, the competitive ratio increases linearly in η (note that the x axis is on log scale). However, if η is large, the competitive ratio remains constant in η at a value of PCR. Note that, for any η, the competitive ratio is always five in the case of zero confidence ($R_1 = r_1$ and $R_2 = r_2$).

Figure 2. (Color online) The analytical performance of ABCS (Algorithm 3). (a) The competitive ratio as a function of the normalized accuracy of the predictions $(\sqrt{2\omega\beta}+\theta)\eta$ for varying values of the confidence r. The competitive ratio increases as predictions are less accurate but remains bounded. (b) The PCR and the OCR as a function of the confidence r. The figure interpolates between the purely online scenario (OCR = PCR = 5) and the machine learning scenario (OCR = 1 and PCR = ∞).



Remark 9. The fact that ABCS achieves a consistency close to one, albeit in a trade-off with robustness, is quite unique. For example, the seminal paper by Lykouris and Vassilvtiskii [38] provides an algorithm in the context of caching that is at most two consistent. Also, Antoniadis et al. [4] provide a nine-consistent deterministic algorithm for the problem of metrical task systems. The randomized algorithm introduced in the same paper is $(1+\varepsilon)$ consistent but has a large additive factor. Corollary 2 shows that ABCS achieves $(1+\varepsilon)$ consistency without any additive factor.

Remark 10. Figure 2(b) plots the values of PCR and OCR as a function of the confidence hyperparameter r. It depicts the interpolation between the purely online scenario (OCR = PCR = 5) and the machine learning scenario (OCR = 1 and PCR = ∞). The current work generalizes these two extremes to any scenario in between. As mentioned in Section 1, we provide performance guarantees for ABCS for any value of the confidence hyperparameter $r \geq 1$. However, the specific choice of r would depend on where the system designer wants to place the system on the red and blue curves in Figure 2(b); view it as a risk-versus-gain curve. For example, the figure shows that if one chooses a value of r so that if the predictions turn out to be accurate, ABCS would be 3 competitive, then that would put the system at the risk of being up to about 18 competitive if the predictions turn out to be completely wrong. Later, in Proposition 3, we show that the trade-off between OCR versus PCR that we obtain for ABCS is necessary in the sense that any algorithm that is $(1+\delta)$ competitive in the optimistic case must be at least $1/(4\delta)$ competitive in the pessimistic case.

Remark 11. Recently, there has been some interest in understanding the performance of algorithms when an estimate of the prediction's accuracy η is available in terms of some probability distribution (Mitzenmacher [44], Mitzenmacher [45]). In such cases, Theorem 4 allows one to calculate the optimal choice of confidence hyperparameter r that minimizes the expected competitive ratio. Assume that the prediction's accuracy η follows some distribution $\mu(\cdot)$. The distribution $\mu(\cdot)$ might, for example, be estimated based on historically observed data. For a fixed r, note that OCR and PCR are functions of r. Denote

$$\zeta(r) := \frac{\text{PCR} - \text{OCR}}{2\text{OCR}}.$$

The expected value of the *random* competitive ratio of ABCS is then

$$\begin{aligned}
\mathbb{E}_{\eta\sim\mu}[\text{CR}(\eta)] &= \int_0^\infty \min((1+2\eta)\cdot\text{OCR}, \text{PCR})\mathrm{d}\mu(\eta) \\
&= 2\text{OCR}\cdot\int_0^{\zeta(r)} \eta\mathrm{d}\mu(\eta) + \text{OCR}\cdot\int_0^{\zeta(r)}\mathrm{d}\mu(\eta) + \text{PCR}\cdot\int_{\zeta(r)}^\infty \mathrm{d}\mu(\eta) \\
&= 2\text{OCR}\cdot\mathbb{E}[\eta\mathbb{1}\{\eta\leq\zeta(r)\}] + \text{OCR}\cdot\mathbb{P}(\eta\leq\zeta(r)) + \text{PCR}\cdot\mathbb{P}(\eta>\zeta(r)). \qquad (4.16)
\end{aligned}$$

Therefore, if either the distribution or an estimate thereof is known, then the parameters of ABCS can be chosen to minimize the expected competitive ratio.

Theorem 4 and Corollary 2 demonstrate that there is a trade-off between the OCR and the PCR. The following proposition shows that this trade-off is, in fact, inherent to the problem and is not an artifact of the algorithm.

Proposition 3. *Let $\mathcal{A}$ be any deterministic algorithm for the capacity-scaling problem in (2.1) and $\text{CR}(\eta)$ denote its competitive ratio when it is has access to an η-accurate prediction. There exist choices of ω, β, and θ such that for any $\delta > 0$, if $\text{CR}(0) \leq 1+\delta$, then*

$$\text{CR}\left(\frac{1}{\delta}\right) \geq \frac{1}{4\delta}. \qquad (4.17)$$

In short, any deterministic algorithm that is $(1+\delta)$ consistent must be $\Omega(1/\delta)$ competitive.

Proposition 3 is proved in Appendix A.7. In comparing Corollary 2 and Proposition 3, one may notice that there is a gap between the consistency-competitiveness trade-off achieved by ABCS and the provable lower bound on this trade-off. Improving the lower-bound result in Proposition 3 or designing an algorithm with a better trade-off is left as an interesting future research direction.

As mentioned earlier, an algorithm for capacity scaling must consist of two components; one component decides when to activate a server, and the other component decides when to deactivate a server. For the latter problem, a popular state-of-the-art approach is to implement a power-down timer (Augustine et al. [6], Gandhi et al. [20], Irani et al. [28],

Karlin et al. [31], Lu et al. [37], Mukherjee et al. [47]). The power-down timer works as follows; each time a server becomes idle, the system starts a timer corresponding to that server. If the server remains idle after the timer expires, then the server is deactivated. Algorithm 4 shows the timer algorithm for any choice of power-down timer $\tau : \mathbb{R}_+^3 \to (0,\infty)$.

Algorithm 4 (The Timer Algorithm (τ))

At each time $t \geq 0$:

Turn off a server if the server has been idle for more than $\tau(\omega,\beta,\theta)$ time.

We end this section by pointing out that, although the timer algorithm has proven to be successful under specific (especially stochastic) scenarios, the worst-case performance of the algorithm in the current context is poor as the following proposition shows. In fact, Proposition 4 shows that there do not exist any choices of ω, β, and θ such that the timer algorithm has a bounded competitive ratio. To the best of our knowledge, there does not exist any competitive algorithm for the capacity-scaling problem where ω is finite, until in the current work. Proposition 4 is proved in Appendix A.8.

Proposition 4. *Let $\tau : \mathbb{R}_+^3 \to (0,\infty)$ be any arbitrary function. For any fixed $\omega,\beta,\theta > 0$, let* CR *be the competitive ratio of the timer algorithm (Algorithm 4) with power-down timer $\tau(\omega,\beta,\theta)$. Then,* $\mathrm{CR} = \infty$.

4.3. Offline Algorithm

We end the main results by providing an approximation algorithm for the offline problem. Although finding an efficient solution to the offline problem is not the main focus of the current paper, the results in this section will be used to run the numerical experiments in Section 5. Also, the algorithm may be used by AP to compute the optimal number of servers given the *predicted* arrival function (see Section 4.2.1). Moreover, the approximation algorithm raises a question about the trade-off between the numerical complexity and the accuracy of the solution, which might be of independent interest.

As a measure of numerical complexity, let us introduce the following regularity assumption on the arrival rate function.

Assumption 1 (Regular). *We say that a function $\lambda : [0,T] \to \mathbb{R}_+$ is δ regular if $\lambda(i\delta + s) = \lambda(i\delta)$ for all $s \in [0,\delta)$ and $i = 0,1,\ldots,\lfloor T/\delta \rfloor$.*

The regularity assumption is a reasonable approximation for any arrival rate function occurring in practice for δ sufficiently small. Now, we state the proposed offline approximation algorithm. For the sake of notation, assume that T is divisible by δ. Let $n = T/\delta$, $q_1 = 0$, and $m_0 = 0$. The offline algorithm solves the following linear program:

$$\begin{aligned} \underset{m,d\in\mathbb{R}^n,q\in\mathbb{R}^{n+1}}{\text{minimize}} \quad & \omega\delta \cdot \sum_{i=1}^{n} \frac{q_i + q_{i+1}}{2} + \beta \cdot \sum_{i=1}^{n} d_i + \theta\delta \cdot \sum_{i=1}^{n} m_i \\ \text{subject to} \quad & q_{i+1} \geq q_i + \int_{(i-1)\delta}^{i\delta} \lambda(t)\mathrm{d}t - \delta m_i \quad \text{for all } i = 1,\ldots,n\,. \\ & d_i \geq m_i - m_{i-1} \quad \text{for all } i = 1,\ldots,n \\ & q_{i+1}, d_i, m_i \geq 0 \quad \text{for all } i = 1,\ldots,n \end{aligned} \tag{4.18}$$

The linear program is a discretization of the problem in (2.1). The vectors q and m represent the workload in the buffer and the number of servers, respectively. To obtain a solution from (4.18) for the original problem in (2.1), set $m(i\delta + s) = m_{i+1}$ for all $s \in [0,\delta)$ and $i = 0,1,\ldots,T/\delta - 1$. Hence, the linear program in (2.1) computes the number of servers that would minimize the cost but where the number of servers is restricted to be δ regular. The constraints and objective value in (2.1) then directly reduce to the constraints and objective value in (4.18). Theorem 5 characterizes the competitive ratio of the solution to the linear program with respect to the offline optimum OPT in (2.3) for any fixed choice of $\delta > 0$. The proof of Theorem 5 is given in Appendix A.3.

Theorem 5. *Let CR denote the competitive ratio of the solution to the linear program (4.18). If λ is δ regular, then*

$$\mathrm{CR} \leq \left(1 + \frac{\omega\delta}{2\theta}\right)\left(1 + \frac{\omega\delta^2}{\beta}\right). \tag{4.19}$$

5. Numerical Experiments

We implemented the algorithms proposed in the current paper and evaluated their performance on both a real-world data set of internet traffic and two artificially generated workloads. The real-world data set consists of DNS requests

observed at a campus network across four consecutive days in April 2016 (Manmeet et al. [41]). The data set represents the typical intensity of internet traffic in this network, and we consider a data center that serves this traffic. The use of internet traffic data sets was common practice in earlier empirical studies (Bamas et al. [8], Gandhi et al. [21]). We let $\lambda(t)$ be the number of requests per second according to this data set. To empirically verify the performance of our algorithms against extreme cases, we also tested the algorithms on two artificially generated arrival rate functions. The arrival rate functions present highly stylized versions of particular patterns that may occur in real-world traffic. Here, we let $\lambda(t)$ be either sinusoidal or a step function. The weights β, θ, and ω are modeled after realistic parameters (Lin et al. [36], Shehabi et al. [53]). In particular, we assume that each server consumes 850 W at a price of 0.15 cents per 1 kWh, β is equal to the power cost of running a server for four hours, and $\omega = 0.1$ cents.

5.1. Purely Online Scenario

We first test the AP, BCS, and timer algorithms that do not require predictions (for the AP algorithm, we let $\tilde{\lambda}(t) = 0$ for all t). The timer algorithm has a threshold of $\tau(\omega, \beta, \theta) = \beta/\theta$ and turns servers on whenever $\lambda(t) > m(t)$, which is typically used in the literature (Augustine et al. [6], Gandhi et al. [20], Irani et al. [28], Karlin et al. [31], Lu et al. [37], Mukherjee et al. [47]). Figure 3 shows the number of servers on the real-world data set and the artificial patterns. Table 1 summarizes the competitive ratios in each scenario.

The performance of BCS on the real-world data set is excellent, only 20% more than the offline optimal solution, even without predictions. On the artificial patterns, BCS is outperformed by both AP and timer algorithms. AP and timer algorithms seem to work well if the data do not contain any sudden spikes of workload. Also, note that the competitive ratio of BCS in both cases is significantly lower than the worst-case competitive ratio of five.

5.2. Machine Learning Scenario

Next, we test the algorithms AP and ABCS in the case that predictions are provided. For the real-world data set, we evaluate three types of predictions.

- Type 1. The system does not reveal any predictions (i.e., $\tilde{\lambda}(t) = 0$ for all $t \in [0, T]$).
- Type 2. The system reports the moving average across three hours (i.e., $\tilde{\lambda}(t) = (\min(t, 1.5) + \min(T - t, 1.5))^{-1} \int_{\max(t-1.5,0)}^{\min(t+1.5,T)} \lambda(t)\mathrm{d}t$ for all $t \in [0, T]$).
- Type 3. The system provides perfect predictions (i.e., $\tilde{\lambda}(t) = \lambda(t)$ for all $t \in [0, T]$).

For the artificial patterns, we evaluate four types of predictions.

- Type 1. The system does not reveal any predictions (i.e., $\tilde{\lambda}(t) = 0$ for all $t \in [0, T]$).
- Type 2. The system predicts only the average of the arrival rate (i.e., $\tilde{\lambda}(t) = 500$ for all $t \in [0, T]$).
- Type 3. The system predicts the opposite of the arrival rate (i.e., $\tilde{\lambda}(t) = 1,000 - \lambda(t)$ for all $t \in [0, T]$).
- Type 4. The system provides perfect predictions (i.e., $\tilde{\lambda}(t) = \lambda(t)$ for all $t \in [0, T]$).

Table 2 summarizes the competitive ratios in each scenario, where ABCS is evaluated for three choices of the confidence hyperparameter (low confidence ($r = 1$), medium confidence ($r = 3$), and high confidence ($r = 5$)).

The performance of ABCS on the real-world data set is excellent. ABCS even reproduces an optimal solution in the case that perfect predictions are available and the confidence is medium or larger. The performance generally

Figure 3. (Color online) The real-world arrival pattern and two artificial arrival patterns considered and the number of servers of OPT, AP, BCS, and timer algorithms. (a) Real world. (b) Sinusoidal. (c) Step function.

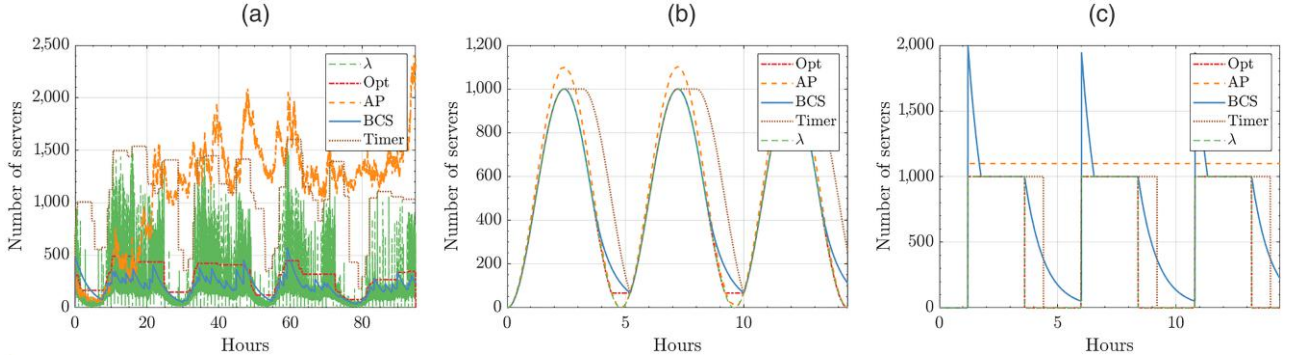


Table 1. The competitive ratios of AP, BCS, and timer algorithms without predictions.

CR	AP	BCS	Timer
Real world	27.7	1.2	3.1
Sinusoidal	1.1	1.4	1.2
Step function	1.8	2.2	1.3

improves as more accurate predictions are available. Moreover, in contrast to AP, ABCS is robust against inaccurate predictions. The competitive ratio of ABCS is close to one for type 1 predictions (and type 3 predictions for the artificial patterns), even in the case of high confidence. Hence, in practice, ABCS is able to reproduce the optimal solution if sufficiently accurate predictions are provided while maintaining a competitive ratio close to one, even if the predictions are completely inaccurate.

6. Proofs

6.1. Proof of Theorem 1

We will provide a high-level overview of the proof of Theorem 1 and refer to the appendix for the details. Recall that BCS is a special case of ABCS (let $R_1 = r_1$, $R_2 = r_2$). To prove Theorem 1, we will, in fact, establish a more general result in Proposition 5, where the rates r_1 and r_2 may vary as rate functions over time. Proposition 5 states that the competitive ratio of ABCS never exceeds PCR irrespective of the magnitude of the error in prediction. Theorem 1 thus follows immediately by letting $R_1 = r_1$ and $R_2 = r_2$.

Proposition 5. *Fix a finite-time horizon T and arrival rate function $\lambda(\cdot)$. Let* Opt *be as defined in* (2.3) *and $m(t)$ be the number of servers of ABCS* (*Algorithm* 3) *when it has access to a prediction $\tilde{\lambda}$. Then,*

$$\textsc{Cost}^{\lambda}(m,T) \le \text{PCR} \cdot \textsc{Opt} \tag{6.1}$$

for all instances (T,λ) and predictions $\tilde{\lambda}$, where PCR is as defined in (4.14).

The proof of Proposition 5 is based on a potential function argument and is provided in Appendix A.9. We end this section by giving a proof sketch of Proposition 5.

Proof Sketch of Proposition 5. Let $m(\cdot)$ be the number of servers of ABCS and $m^*(\cdot)$ be a differentiable optimal solution to the offline optimization Problem (2.1). Appendix A.9 shows how to extend this to arbitrary nondifferentiable solutions. Let $\Phi(\cdot)$ be a nonnegative potential function such that

$$\frac{\mathrm{d}\Phi(t)}{\mathrm{d}t} + \frac{\partial \textsc{Cost}^{\lambda}(m,t)}{\partial t} \le \text{PCR} \cdot \frac{\partial \textsc{Cost}^{\lambda}(m^*,t)}{\partial t} \tag{6.2}$$

and $\Phi(0) = 0$, assuming, for now, that such a $\Phi(\cdot)$ exists. We integrate Equation (6.2) from time $t = 0$ to $t = T$ to obtain

$$\textsc{Cost}^{\lambda}(m,T) \le \text{PCR} \cdot \textsc{Cost}^{\lambda}(m^*,T) + \Phi(0) - \Phi(T) \le \text{PCR} \cdot \textsc{Cost}^{\lambda}(m^*,T), \tag{6.3}$$

Table 2. The competitive ratios of AP and ABCS for different values of the confidence hyperparameter in the presence of predictions.

CR(η)	AP	ABCS (low confidence)	ABCS (medium confidence)	ABCS (high confidence)
Real world				
Type 1	27.7	1.19	1.00	1.00
Type 2	14.2	1.19	1.07	1.38
Type 3	1.00	1.19	1.00	1.00
Sinusoidal				
Type 1	1.10	1.43	1.16	1.12
Type 2	1.21	1.43	1.18	1.14
Type 3	1.46	1.43	1.17	1.17
Type 4	1.00	1.43	1.11	1.15
Step function				
Type 1	1.85	2.17	1.68	1.64
Type 2	1.72	2.17	1.61	1.56
Type 3	1.90	2.17	1.68	1.63
Type 4	1.00	2.17	1.37	1.15

where the last step follows because $\Phi(T)$ is nonnegative and $\Phi(0) = 0$. The proof of Proposition 5 is, therefore, completed if we manage to find a potential function $\Phi(t)$ satisfying Equation (6.2) and $\Phi(0) = 0$. Define the potential function $\Phi(t)$ such that

$$\Phi(t) = \begin{cases} c_5\beta \cdot \left(d_{R_1}(t) - m(t) + m^*(t)\right), & \text{if } m(t) > m^*(t) \\ c_6\beta \cdot \left(d_{r_1}(t) - m(t) + m^*(t)\right) & \text{if } m(t) \le m^*(t) \end{cases} + \frac{\beta \cdot m(t)}{r_2} + c_6 R_2 \theta \cdot [q(t) - q^*(t)]^+, \tag{6.4}$$

where c_5 and c_6 are as defined in Equation (4.14) and

$$d_r(t) = \sqrt{\frac{r\omega \cdot ([q(t) - q^*(t)]^+)^2}{\beta} + (m(t) - m^*(t))^2}. \tag{6.5}$$

Some intuitions on the construction of this potential function are given in Section 6.2. Note that $\Phi(t)$ is nonnegative and $\Phi(0) = 0$. It remains to show that $\Phi(t)$ satisfies Equation (6.2), which profoundly relies on $d_r(t)$. The full argument involves a case distinction and is provided in Appendix A.9. For this proof sketch, let us only consider the case that $m(t) > m^*(t)$, $q(t) > q^*(t)$, $\frac{\mathrm{d}m}{\mathrm{d}t} \ge 0$, and $\frac{\mathrm{d}m^*}{\mathrm{d}t} \ge 0$. Recall that by the definition of ABCS,

$$\frac{\mathrm{d}q(t)}{\mathrm{d}t} = \lambda(t) - m(t), \frac{\mathrm{d}m(t)}{\mathrm{d}t} = \frac{\hat{r}_1(t)\omega \cdot q(t) - \hat{r}_2(t)\theta \cdot m(t)}{\beta} \le \frac{R_1\omega \cdot q(t)}{\beta}. \tag{6.6}$$

The derivative of $d_{R_1}(t)$ is at most

$$\begin{aligned} \beta \cdot \frac{\mathrm{d}d_{R_1}}{\mathrm{d}t} &\le d_{R_1}(t)^{-1} \cdot \begin{pmatrix} R_1\omega \cdot (q(t) - q^*(t))(\lambda(t) - m(t)) \\ + R_1\omega \cdot (q^*(t) - q(t))(\lambda(t) - m^*(t)) \\ + R_1\omega \cdot q(t) \cdot (m(t) - m^*(t)) \\ + \beta \cdot \frac{\mathrm{d}m^*}{\mathrm{d}t} \cdot (m^*(t) - m(t)) \end{pmatrix} \\ &= d_{R_1}(t)^{-1} \cdot (m(t) - m^*(t)) \left(R_1\omega \cdot q^*(t) - \beta \cdot \frac{\mathrm{d}m^*}{\mathrm{d}t}\right) \le R_1\omega \cdot q^*(t). \end{aligned} \tag{6.7}$$

Crucially, the derivative does not contain any terms involving $m(t)$ or $q(t)$ but only $q^*(t)$, which is easy to bound by the cost of the optimal solution. Thus, the derivative of $\Phi(t)$ can be upper bounded as follows:

$$\frac{\mathrm{d}\Phi(t)}{\mathrm{d}t} \le c_5 R_1\omega \cdot q^*(t) - \left(1 + \frac{1}{r_1}\right)\beta \cdot \frac{\mathrm{d}m}{\mathrm{d}t} + c_5\beta \cdot \frac{\mathrm{d}m^*}{\mathrm{d}t} + \left(1 + \frac{R_2}{r_1}\right)\theta \cdot (m^*(t) - m(t)), \tag{6.8}$$

where the constants c_5 and c_6 have been expanded according to their definitions in (4.14). Observe that the derivative of the potential function $\Phi(t)$ exactly cancels the cost of ABCS because $\omega \cdot q(t) \le \frac{\beta}{r_1} \cdot \frac{\mathrm{d}m(t)}{\mathrm{d}t} + \frac{R_2\theta m(t)}{r_1}$. We, therefore, obtain

$$\begin{aligned} \frac{\mathrm{d}\Phi(t)}{\mathrm{d}t} + \frac{\partial \textsc{Cost}^\lambda(m,t)}{\partial t} &\le c_5 R_1\omega \cdot q^*(t) + c_5\beta \cdot \frac{\mathrm{d}m^*}{\mathrm{d}t} + \left(1 + \frac{R_2}{r_1}\right)\theta \cdot m^*(t) \\ &\le \max\left(c_5 R_1, c_5, 1 + \frac{R_2}{r_1}\right) \cdot \frac{\partial \textsc{Cost}^\lambda(m^*,t)}{\partial t}. \end{aligned} \tag{6.9}$$

Note that removing the term $d_r(t)$ from $\Phi(t)$ would yield a similar form as Equation (6.8). However, the resulting potential function is not nonnegative, hence the need for the term $d_r(t)$. □

6.2. Intuition on the Potential Function

The potential function is the most important ingredient in the proof of Proposition 5, which then boils down to verification of (6.2). We discuss the intuition behind the choice of the potential function $\Phi(t)$ here.

First, from a high-level perspective, the terms of this function are chosen such that their derivative cancels the terms in the derivative of the cost of ABCS in verification of (6.2). The term $c_6 R_2 \theta q(t)$ is designed to bound the

power consumption of ABCS. The derivative of this term equals $-c_6 R_2 \theta m(t)$, which cancels the power consumption $\theta m(t)$ of ABCS. The term $-(c_i - 1/r_2)\beta m(t)$ is designed to bound the switching cost (for $i = 5, 6$). The derivative of this term equals $-(c_i - 1/r_2)\beta \frac{dm}{dt}$, which cancels the switching cost $\beta \frac{dm}{dt}$ of ABCS. The waiting cost is the most subtle to bound because it cannot be written as the derivative of a first-order property of the system. It follows by the property of ABCS that $\omega q(t) \le \frac{\beta}{r_1} \cdot \frac{\mathrm{d}m(t)}{\mathrm{d}t} + \frac{R_2 \theta m(t)}{r_1}$. In words, the waiting cost is bounded by a weighted sum of the switching cost and the power consumption. The choice of the constants, therefore, ensures that the derivative of potential function also cancels the waiting cost of ABCS. Finally, the term $d_r(t)$ is required to ensure that $\Phi(t)$ is nonnegative. Although we need a term of the form $m^*(t) - m(t)$, we cannot simply add this to the potential function because this can make the potential function negative if $m(t) > m^*(t)$. The term $d_r(t)$ ensures that the potential function is nonnegative, and this particular choice of $d_r(t)$ also guarantees that the derivative is bounded by quantities of the optimal solution, such as $q^*(t)$ (see, for example, (6.7)).

Quantitatively, the term $d_r(t)$ consists of the difference $[q(t) - q^*(t)]^+$, which is zero if $q(t) \le q^*(t)$ and strictly positive otherwise. Clearly, if $q(t) > q^*(t)$, then ABCS needs to spend more in the future to reduce the additional workload. The workload difference is enclosed in an ℓ_2 metric together with the difference in the number of servers. This is the crux of the potential function and leads to a diminishing marginal penalty of the difference in queue lengths. The larger the difference $m(t) - m^*(t)$, the less influence a unit increase of $[q(t) - q^*(t)]^+$ has on the potential function. Hence, the penalty of the queue-length difference is measured relative to the difference in the number of servers, which is natural and essential in the proof.

6.3. Proof of Theorem 4

Theorem 4 states that the competitive ratio is at most the minimum of the OCR and the PCR. Proposition 5 in the previous section showed that the competitive ratio of ABCS is at most PCR. To prove the bound on the competitive ratio by OCR, we will relate the performance of ABCS to the cost achieved by the subroutine AP. Proposition 6 states that the cost of ABCS differs by at most a factor of OCR from the cost of AP.

Proposition 6. *Fix a finite-time horizon T and arrival rate function $\lambda(\cdot)$. Let* Opt *be as defined in* (2.3) *and $m(t)$ be the number of servers of BCS* (*Algorithm* 1) *when it has access to a prediction $\tilde{\lambda}$. Then,*

$$\textsc{Cost}^{\lambda}(m, T) \le \mathrm{OCR} \cdot \textsc{Cost}^{\lambda}(\tilde{m}, T) \tag{6.10}$$

for all instances (T, λ) and predictions $\tilde{\lambda}$, where $\tilde{m}(t)$ represents the advised number of servers of AP and OCR *is as defined in* (4.14).

As already hinted at in Remark 11, Proposition 6 is independent of the source of the advice and holds for any function $\tilde{m}(\cdot)$. Therefore, if there exists another algorithm that provides advice besides AP, then this advice may be readily used in ABCS, and the competitive ratio of ABCS with respect to the new advice is again at most OCR. The proof of Proposition 6 is based on a potential function argument and is provided in Appendix A.10. The proof follows a similar structure as the proof of Proposition 5. We now have all the ingredients to prove Theorem 4.

Proof of Theorem 4. The proof follows almost immediately from Propositions 5 and 6. Note that $\mathrm{CR}(\eta) \le \mathrm{PCR}$ because

$$\textsc{Cost}^{\lambda}(m, T) \le \mathrm{PCR} \cdot \textsc{Opt} \tag{6.11}$$

by Proposition 5. Next, $\mathrm{CR}(\eta) \le (1 + (\sqrt{2\omega\beta} + \theta)\eta) \cdot \mathrm{OCR}$ because

$$\textsc{Cost}^{\lambda}(m, T) \le \mathrm{OCR} \cdot \textsc{Cost}^{\lambda}(\tilde{m}, T) \le (1 + (\sqrt{2\omega\beta} + \theta)\eta) \cdot \mathrm{OCR} \cdot \textsc{Opt} \tag{6.12}$$

by Proposition 6 and Theorem 3. □

7. Conclusion

In this paper, we explored how ML predictions can be used to improve the performance of capacity-scaling solutions without sacrificing robustness. We extend the state of the art in capacity scaling by analyzing a more general model in continuous time where tasks are allowed to wait, in which case popular earlier proposed algorithms are not competitive. The BCS algorithm we proposed is five competitive in the general case. We also introduced AP, which is one competitive if the ML predictions are accurate. Finally, we proposed ABCS, which combines the ideas behind BCS and AP. We proved that in the presence of accurate ML predictions, ABCS is $(1 + \varepsilon)$ competitive and that its performance degrades gracefully in the prediction's accuracy. Moreover, the competitive ratio of ABCS is at most $\mathcal{O}(\varepsilon^{-7/2})$ when ML predictions are inaccurate. Although the competitive ratio of ABCS depends on the accuracy, the algorithm is oblivious to it. In the context of data centers, because real-world internet traffic is erratic, any implemented capacity-

scaling solution must be robust against sudden unpredictable surges in workload. Our results yield significant reductions in power consumption while maintaining robustness against these sudden spikes.

An interesting yet challenging direction for future work is to restrict the number of servers to be integer valued. A common approach is to randomly round a fractional solution, such as the one returned by ABCS, and prove that the increase in competitive ratio because of the random rounding is at most a constant factor in expectation. Apart from the fact that one cannot expect $(1+\varepsilon)$ consistency by this procedure, the integrality gap is in fact unbounded for this problem, as shown by Lemma 2. The proof of Lemma 2 is provided in Appendix A.11.

Lemma 2. *Let* $\textsc{Opt}_{int}$ *be the offline minimum cost where* $m(\cdot)$ *is restricted to be integer valued: that is,*

$$\textsc{Opt}_{int} := \inf_{m:(0,T]\to\mathbb{N}} \textsc{Cost}^{\lambda}(m,T). \tag{7.1}$$

Also, let CR denote the competitive ratio of $\textsc{Opt}_{int}$ *with respect to* Opt *in* (2.3). *Then,* $\mathrm{CR}=\infty$.

Lemma 2 forbids any rounding procedure from being competitive, and we, therefore, leave the integral case as an open question. Finally, in an ongoing work, we are exploring how the confidence hyperparameter of ABCS can be learned over time if there are statistical guarantees on the prediction's accuracy.

Acknowledgments
The authors thank Ravi Kumar (Google) for inspiring discussions that started this project.

Appendix. Proofs
This section provides the proofs that have been omitted from the main text.

A.1. Proof of Proposition 1

Proof of Proposition 1. Fix a finite-time horizon T, arrival rate function $\lambda(\cdot)$, and initial number of servers $m(0)$. Let

$$V^{+}(f)=\limsup_{\delta\downarrow 0}\sum_{i=0}^{\lfloor T/\delta\rfloor}[f(i\delta+\delta)-f(i\delta)]^{+} \tag{A.1}$$

for a function $f:[0,T]\to\mathbb{R}$. The definition of $V^{+}(f)$ is closely related to the notion of bounded variation. The bounded variation of a function $f:[0,T]\to\mathbb{R}$ is defined as

$$V(f)=\sup\left\{\sum_{i=1}^{n}|f(z_i)-f(z_{i-1})| \text{ such that } \{z_i\}_{i=0}^{n} \text{ is a partition of } [0,T]\right\}. \tag{A.2}$$

Let m_n be a sequence of functions such that $\textsc{Cost}^{\lambda}(m_n,T)\to\textsc{Opt}$ as $n\to\infty$. There exists $N\in\mathbb{N}$ such that $\textsc{Cost}^{\lambda}(m_n,T)\le 2\cdot\textsc{Opt}$ for all $n\ge N$. As a result, $V^{+}(m_n)$ is uniformly bounded for $n\ge N$. Note that without increasing the cost, we can set $m_n(T)=0$. The bounded variation and $V^{+}(m_n)$ are then related as

$$V(m_n)=2V^{+}(m_n)+m(0). \tag{A.3}$$

The rest of the proof depends on the following compactness theorem (Barbu and Precupanu [10]).

Theorem A.1 (Helly's Selection Theorem). *Let* $f_n:[0,T]\to\mathbb{R}$ *be a sequence of functions, and suppose that the next two conditions hold.*

i. *The sequence* f_n *has uniformly bounded variation* (*i.e.*, $\sup_{n\in\mathbb{N}}V(f_n)<\infty$).
ii. *The sequence* f_n *is uniformly bounded at a point* (*i.e., there exists* $t\in[0,T]$ *such that* $\{f_n(t)\}_{n=1}^{\infty}$ *is a bounded set*).

Then, there exists a subsequence f_{n_k} *of* f_n *and a function* $f:[0,T]\to\mathbb{R}$ *such that*

i. f_{n_k} *converges to* f *pointwise as* $k\to\infty$,
ii. f_{n_k} *converges to* f *in* L_1 *as* $k\to\infty$, *and*
iii. $V(f)\le\liminf_{k\to\infty}V(f_{n_k})$.

Recall the infinite sequence $m_N, m_{N+1},\dots$ introduced. Condition (i) in Theorem A.1 holds because

$$V(m_n)=2V^{+}(m_n)+m(0)\le\frac{4\textsc{Opt}}{\beta}+m(0) \tag{A.4}$$

for all $n\ge N$. Moreover, condition (ii) in Theorem A.1 holds for $t=0$ because $m_n(0)=m(0)$ for all $n\in\mathbb{N}$. Hence, there exists a subsequence m_{n_k} of m_n and a function $m^{*}:[0,T]\to\mathbb{R}$ such that $m_{n_k}\to m^{*}$ pointwise and in the L_1 norm, as $k\to\infty$, and

$$V^{+}(m^{*})=(V(m^{*})-m(0))/2\le\liminf_{k\to\infty}(V(m_{n_k})-m(0))/2=\liminf_{k\to\infty}V^{+}(m_{n_k}). \tag{A.5}$$

Therefore, because the flow time and the power cost are continuous in m with respect to the L_1 norm,

$$\begin{aligned}\textsc{Cost}^{\lambda}(m^*,T) &= \omega\cdot\int_0^T q^*(t)\mathrm{d}t+\beta\cdot V^+(m^*)+\theta\cdot\int_0^T m^*(t)\mathrm{d}t\\ &\le \omega\cdot\lim_{k\to\infty}\int_0^T q_{n_k}(t)\mathrm{d}t+\beta\cdot\liminf_{k\to\infty}V^+(m_{n_k})+\theta\cdot\lim_{k\to\infty}\int_0^T m_{n_k}(t)\mathrm{d}t\\ &\le \lim_{k\to\infty}\textsc{Cost}^{\lambda}(m_{n_k},T)=\textsc{Opt},\end{aligned}\tag{A.6}$$

which completes the proof of the proposition. □

A.2. Proof of Lemma 1

Proof of Lemma 1. Fix any online algorithm $\mathcal{A}$ and parameters ω and β. We will construct an instance for which $\textsc{Cost}^{\lambda}(m,T)\ge\frac{\omega t_0^2}{2\beta}\cdot\textsc{Opt}$.

Let $\lambda(t)=0$ for $t\in[0,t_0]$ and $\lambda(t)=\rho$ for $t\in(t_0,2t_0]$, where the value of ρ will be chosen (adversarially) later. Let the finite-time horizon $T=2t_0$ and $\theta=0$. Let $m(t)$ be the number of servers of $\mathcal{A}$ for the instance. We distinguish two cases depending on $\overline{m}:=\sup_{t\in(t_0,2t_0]}m(t)$. Crucially, note that any decision to increase $m(t)$ during the interval $[t_0,2t_0]$ must be taken during the interval $[0,t_0]$ because t_0 is the setup time.

1. First, consider the case when $\overline{m}>0$. The increment in $m(t)$ must have been initiated during $[0,t_0]$. In that case, fix $\rho=0$. One possible solution of (2.1) is $m^*(t)=0$ for $t\in[0,T]$, and the cost of the optimal solution is, therefore, at most $\textsc{Opt}=0$. However, the cost of $\mathcal{A}$ is at least $\textsc{Cost}^{\lambda}(m,T)\ge\beta\overline{m}>0$. Therefore, there does not exist a constant CR such that $\textsc{Cost}^{\lambda}(m,T)\le\mathrm{CR}\cdot\textsc{Opt}$ and hence, $\mathrm{CR}=\infty$ by definition.
2. Next, consider the case when $\overline{m}=0$ (i.e., no increment in $m(t)$ has been initiated during $[0,t_0]$). In that case, fix $\rho=1$. One possible solution of (2.1) is $m^*(t)=1$ for $t\in[0,T]$. This solution does not incur any waiting cost, and the cost of the optimal solution is, therefore, at most $\textsc{Opt}\le\beta$. However, the cost of $\mathcal{A}$ is $\textsc{Cost}^{\lambda}(m,T)=\omega\cdot\int_0^{t_0}\rho s\,\mathrm{d}s=\frac{\omega t_0^2}{2}\ge\frac{\omega t_0^2}{2\beta}\cdot\textsc{Opt}$.

This completes the proof of the lemma. □

A.3. Proof of Theorem 5

Proof of Theorem 5. Fix a finite-time horizon T, arrival rate function $\lambda(\cdot)$, and initial number of servers $m(0)$. Assume that T is divisible by δ, and let $n=T/\delta$. Let

$$\mathcal{C}:=\{f:[0,T]\to\mathbb{R}_+\,|\,f(i\delta+s)=f(i\delta)\text{ for all }s\in[0,\delta)\text{ and }i=0,1,\dots,n-1\}\tag{A.7}$$

be the subspace of the space of functions that are constant in each δ interval. Recall that by assumption, $\lambda\in\mathcal{C}$. We note that each $f\in\mathcal{C}$ is equivalently represented by a vector $f=(f(0),f(\delta),\dots,f(n-1))\in\mathbb{R}^n$ and vice versa. We will, therefore, interchangeably use vector notation to denote an element from $\mathcal{C}$.

Claim A.1. We claim that

$$\inf_{m\in\mathcal{C}}\textsc{Cost}^{\lambda}(m,T)\le\left(1+\frac{\omega\delta^2}{\beta}\right)\inf_{m:[0,T]\to\mathbb{R}_+}\textsc{Cost}^{\lambda}(m,T)+\frac{\omega\delta^2\cdot m(0)}{2}.\tag{A.8}$$

Proof. Let $m^*:[0,T]\to\mathbb{R}_+$ be arbitrary, and let $m_i=\frac{1}{\delta}\int_{(i-1)\delta}^{i\delta}m^*(t)\mathrm{d}t$. We will prove that

$$\textsc{Cost}^{\lambda}(m,T)\le\left(1+\frac{\omega\delta^2}{\beta}\right)\textsc{Cost}^{\lambda}(m^*,T)+\frac{\omega\delta^2\cdot m(0)}{2},\tag{A.9}$$

which finishes the proof of the claim. Note that it follows immediately by construction that the switching cost of m is at most the switching cost of m^* and that the power cost of m is equal to the power cost of m^*. We will, therefore, focus on the flow-time cost. The queue length of m^* at the end points of each δ interval is at least the queue length of m as follows:

$$\begin{aligned}q^*(i\delta)&=q^*((i-1)\delta)+\int_{(i-1)\delta}^{i\delta}\Big(\lambda_i-m^*(s)\Big)\mathbb{1}\{q^*(s)>0\text{ or }\lambda_i\ge m^*(s)\}\mathrm{d}s\\ &\ge\left[q^*((i-1)\delta)+\int_{(i-1)\delta}^{i\delta}\Big(\lambda_i-m^*(s)\Big)\mathrm{d}s\right]^+\\ &\ge[q((i-1)\delta)+\delta\lambda_i-\delta m_i]^+=q(i\delta),\end{aligned}\tag{A.10}$$

where the inequality $q^*((i-1)\delta)\ge q((i-1)\delta)$ follows by induction on i. Define

$$\Delta_i=\sup_{t\in[(i-1)\delta,i\delta]}m^*(t)-\inf_{t\in[(i-1)\delta,i\delta]}m^*(t),\tag{A.11}$$

and observe that

$$\sum_{i=1}^{n}\Delta_i \le m(0) + 2\lim_{\varepsilon\downarrow 0}\sum_{i=0}^{\lfloor T/\varepsilon\rfloor}[m^*(i\varepsilon+\varepsilon) - m^*(i\varepsilon)]^+ \le m(0) + \frac{2\textsc{Cost}^{\lambda}(m^*,T)}{\beta}. \tag{A.12}$$

Then, the flow-time cost of m^* in each δ interval is at least

$$\begin{aligned}\int_{(i-1)\delta}^{i\delta} q^*(t)\mathrm{d}t &= \int_{(i-1)\delta}^{i\delta}\left[q^*((i-1)\delta) + \int_{(i-1)\delta}^{t}(\lambda_i - m(s))\mathbb{1}\{q^*(s)>0 \text{ or } \lambda_i \ge m(s)\}\mathrm{d}s\right]\mathrm{d}t\\ &\ge \int_{(i-1)\delta}^{i\delta}\left[q^*((i-1)\delta) + \int_{(i-1)\delta}^{t}(\lambda_i - m(s))\mathrm{d}s\right]^+\mathrm{d}t\\ &\ge \int_{(i-1)\delta}^{i\delta}\left[q((i-1)\delta) + \int_{(i-1)\delta}^{t}(\lambda_i - m_i - \Delta_i)\mathrm{d}s\right]^+\mathrm{d}t\\ &= \int_{(i-1)\delta}^{i\delta}[q(t) - (t-(i-1)\delta)\Delta_i]^+\mathrm{d}t \ge \int_{(i-1)\delta}^{i\delta} q(t)\mathrm{d}t - \frac{\delta^2\Delta_i}{2},\end{aligned} \tag{A.13}$$

where the second inequality uses (A.10). Therefore,

$$\omega\cdot\int_0^T q(t)\mathrm{d}t - \omega\cdot\int_0^T q^*(t)\mathrm{d}t \le \frac{\omega\delta^2}{2}\cdot\sum_{i=1}^{n}\Delta^i \le \frac{\omega\delta^2\cdot m(0)}{2} + \frac{\omega\delta^2\cdot\textsc{Cost}^{\lambda}(m^*,T)}{\beta}, \tag{A.14}$$

where the second inequality follows by (A.12). This completes the proof of the claim. □

Let $\textsc{Obj}^{\lambda}(m,T)$ denote the value of the objective in (4.18) for $m\in\mathcal{C}$.

Claim A.2. We claim that

$$\begin{aligned}\textsc{Obj}^{\lambda}(m,T) &= \textsc{Cost}^{\lambda}(m,T) + \omega\cdot\sum_{i=1}^{n}\left(\delta\cdot\frac{q_i}{2} - \frac{q_i^2}{2(m_i-\lambda_i)}\right)\mathbb{1}\{q_i>0 \text{ and } q_{i+1}=0\}\\ &\le \left(1+\frac{\omega\delta}{2\theta}\right)\textsc{Cost}^{\lambda}(m,T)\end{aligned} \tag{A.15}$$

for any $m\in\mathcal{C}$.

Proof. Let $m\in\mathcal{C}$ be arbitrary. Note that because $m\in\mathcal{C}$, the switching cost is $\sum_{i=1}^{n}[m_i - m_{i-1}]^+$, and the power cost is $\sum_{i=1}^{n}\delta m_i$, which matches the terms in $\textsc{Obj}^{\lambda}(m,T)$. We will, therefore, focus on the flow-time cost. Denote $q=(q(0),q(\delta),\dots,q(n))\in\mathbb{R}^{n+1}$. The flow time is equal to

$$\begin{aligned}\int_{(i-1)\delta}^{i\delta} q(t)\mathrm{d}t &= \int_0^{\delta}[q_i + (\lambda_i - m_i)t]^+\mathrm{d}t\\ &= \delta\cdot\frac{q_i+q_{i+1}}{2}\mathbb{1}\{q_i=0 \text{ or } q_{i+1}>0\} + \frac{q_i^2}{2(m_i-\lambda_i)}\mathbb{1}\{q_i>0 \text{ and } q_{i+1}=0\}\end{aligned} \tag{A.16}$$

because $q(\cdot)$ increases or decreases linearly. This completes the equality in the claim. To see why the inequality holds, note that

$$\begin{aligned}\sum_{i=1}^{n}\left(\delta\cdot\frac{q_i}{2} - \frac{q_i^2}{2(m_i-\lambda_i)}\right)\mathbb{1}\{q_i>0 \text{ and } q_{i+1}=0\} &\le \frac{\delta}{2}\cdot\sum_{i=1}^{n} q_i\mathbb{1}\{q_i>0 \text{ and } q_{i+1}=0\}\\ &\le \frac{\delta}{2}\cdot\sum_{i=1}^{n}\delta m_i, \le \frac{\delta\cdot\textsc{Cost}^{\lambda}(m,T)}{2\theta},\end{aligned} \tag{A.17}$$

where the second inequality follows because $\delta(m_i-\lambda_i)\ge q_i$. This completes the proof of the claim. □

We now finish the proof of Theorem 5. Let $m\in\mathcal{C}$ be an optimal solution to (4.18). Moreover, define

$$m^* = \arg\min_{m\in\mathcal{C}}\textsc{Cost}^{\lambda}(m,T). \tag{A.18}$$

Then, the cost of m is at most

$$\begin{aligned}\textsc{Cost}^{\lambda}(m,T) &\le \textsc{Obj}^{\lambda}(m,T) \le \textsc{Obj}^{\lambda}(m^*,T)\\ &\le \left(1+\frac{\omega\delta}{2\theta}\right)\textsc{Cost}^{\lambda}(m^*,T)\\ &\le \left(1+\frac{\omega\delta}{2\theta}\right)\left(\left(1+\frac{\omega\delta^2}{\beta}\right)\textsc{Opt} + \frac{\omega\delta^2 m(0)}{2}\right),\end{aligned} \tag{A.19}$$

which completes the proof of the theorem. □

A.4. Proof of Theorem 2

Proof of Theorem 2. Fix a finite-time horizon T, arrival rate function $\lambda(\cdot)$, and initial number of servers $m(0)$. Let $m^*(t)$ be a solution of the offline optimization Problem (2.1) and $m(t)$ be the number of servers of BCS (Algorithm 1). We will prove that

$$\textsc{Cost}(m,T) \le 2\cdot\textsc{Cost}(m^*,T)+\beta\cdot m(0), \tag{A.20}$$

where we have omitted λ from the notation $\textsc{Cost}^{\lambda}(m,T)$.

A.4.1. Overview of the Proof. Let $t_1 \le t_2 \le \dots$ be a partitioning of the interval $[0,T]$ such that (i) $m(t)$ is monotone in $[t_k,t_{k+1}]$ and (ii) either $m(t) > m^*(t)$ or $m(t) \le m^*(t)$ in $[t_k,t_{k+1}]$ for all $k \in \mathbb{N}$. The goal of the proof will be to find a nonnegative potential function $\Phi(t)$ such that

$$\Phi(t_{k+1})-\Phi(t_k)+\textsc{Cost}(m,t_{k+1})-\textsc{Cost}(m,t_k) \le 2\cdot(\textsc{Cost}(m^*,t_{k+1})-\textsc{Cost}(m^*,t_k)) \tag{A.21}$$

for all $k \in \mathbb{N}$. We sum Equation (A.21) over $k \in \mathbb{N}$ to obtain

$$\textsc{Cost}(m,T) \le 2\cdot\textsc{Cost}(m^*,T)+\Phi(0)-\Phi(T) \le 2\cdot\textsc{Cost}(m^*,T)+\Phi(0), \tag{A.22}$$

where the last step follows because $\Phi(T)$ is nonnegative. The proof of Theorem 2 is, therefore, completed if we manage to find a nonnegative potential function $\Phi(t)$ satisfying Equation (A.21) and $\Phi(0)=\beta\cdot m(0)$.

A.4.2. Choice of $\Phi(t)$. Define the potential function $\Phi(t)$ such that

$$\Phi(t)=\begin{cases}\beta\cdot m(t), & \text{if } m(t)>m^*(t)\\ 2\beta\cdot m^*(t)-\beta\cdot m(t) & \text{if } m(t)\le m^*(t)\end{cases}. \tag{A.23}$$

Note that $\Phi(t)$ is nonnegative and $\Phi(0)=\beta\cdot m(0)$.

A.4.3. Verification of (A.21). We continue by verifying Equation (A.21). Fix $k \in \mathbb{N}$. We distinguish two cases depending on whether $m(t)$ is decreasing or nondecreasing in $[t_k,t_{k+1}]$.

i. Assume that $m(s)$ is decreasing for $s \in [t_k,t_{k+1}]$. Recall that by definition,

$$\frac{\mathrm{d}m(t)}{\mathrm{d}t}=-\frac{\theta\cdot m(t)}{\beta} \tag{A.24}$$

for $t \in [t_k,t_{k+1}]$ and therefore,

$$m(t_k+s)=m(t_k)\cdot\exp\left(-\frac{\theta\cdot s}{\beta}\right) \tag{A.25}$$

for $s \in [0,t_{k+1}-t_k]$; hence,

$$\theta\cdot\int_{t_k}^{t_{k+1}} m(s)\mathrm{d}s=\beta\cdot m(t_k)\left(1-\exp\left(-\frac{\theta\cdot(t_{k+1}-t_k)}{\beta}\right)\right)=\beta\cdot(m(t_k)-m(t_{k+1})). \tag{A.26}$$

We further distinguish two cases depending on whether $m(t) > m^*(t)$ or $m(t) \le m^*(t)$ in $[t_k,t_{k+1}]$. First, consider the case that $m(s) > m^*(s)$ for $s \in [t_k,t_{k+1}]$. Then,

$$\begin{aligned}&\Phi(t_{k+1})-\Phi(t_k)+\textsc{Cost}(m,t_{k+1})-\textsc{Cost}(m,t_k)\\ &=\beta\cdot(m(t_{k+1})-m(t_k))+\beta\cdot(m(t_k)-m(t_{k+1}))\\ &=0\le 2(\textsc{Cost}(m^*,t_{k+1})-\textsc{Cost}(m^*,t_k)).\end{aligned} \tag{A.27}$$

Next, consider the case that $m(s) \le m^*(s)$ for $s \in [t_k,t_{k+1}]$. Then,

$$\begin{aligned}&\Phi(t_{k+1})-\Phi(t_k)+\textsc{Cost}(m,t_{k+1})-\textsc{Cost}(m,t_k)\\ &=2\beta\cdot(m^*(t_{k+1})-m^*(t_k))-\beta\cdot(m(t_{k+1})-m(t_k))+\beta\cdot(m(t_k)-m(t_{k+1}))\\ &=2\beta\cdot(m^*(t_{k+1})-m^*(t_k))+2\theta\cdot\int_{t_k}^{t_{k+1}} m(s)\,\mathrm{d}s\\ &\le 2\beta\cdot(m^*(t_{k+1})-m^*(t_k))+2\theta\cdot\int_{t_k}^{t_{k+1}} m^*(s)\,\mathrm{d}s\\ &\le 2(\textsc{Cost}(m^*,t_{k+1})-\textsc{Cost}(m^*,t_k)).\end{aligned} \tag{A.28}$$

ii. Assume that $m(s)$ is nondecreasing for $s \in [t_k,t_{k+1}]$. Note that because tasks are not allowed to wait, $m^*(t) \ge \lambda(t)$ for all $t \in [0,T]$. Recall that by definition, if the arrival rate $\lambda(t)$ is higher than the number of servers $m(t)$, then BCS increases the

number of servers to match the arrival rate. Therefore, $m(s) = \lambda(s)$ for $s \in [t_k, t_{k+1}]$ because $m(t)$ is nondecreasing in $[t_k, t_{k+1}]$. Hence, $m^*(s) \geq m(s) = \lambda(s)$ for $s \in [t_k, t_{k+1}]$ and

$$\begin{aligned}
&\Phi(t_{k+1}) - \Phi(t_k) + \textsc{Cost}(m, t_{k+1}) - \textsc{Cost}(m, t_k) \\
&= 2\beta \cdot (m^*(t_{k+1}) - m^*(t_k)) - \beta \cdot (m(t_{k+1}) - m(t_k)) \\
&\quad + \beta \cdot (m(t_{k+1}) - m(t_k)) + \theta \cdot \int_{t_k}^{t_{k+1}} m(s)\,\mathrm{d}s \\
&\leq 2\beta \cdot (m^*(t_{k+1}) - m^*(t_k)) + \theta \cdot \int_{t_k}^{t_{k+1}} m^*(s)\,\mathrm{d}s \\
&\leq 2(\textsc{Cost}(m^*, t_{k+1}) - \textsc{Cost}(m^*, t_k)).
\end{aligned} \tag{A.29}$$

□

A.5. Proof of Proposition 2

Proof of Proposition 2. Fix any algorithm $\mathcal{A}$, and let $m(t)$ denote its number of servers. We will construct an instance (T, λ), for which $\textsc{Cost}^{\lambda}(m, T) \geq 2.549 \cdot \textsc{Opt}$.

Let $\lambda(t) = 1$ for $t \in [0, T]$. The time horizon T will be specified later. Fix $\beta = \omega = 1$ and $\theta = 0$. Let the prediction $\tilde{\lambda}(t) = 0$ for all t and as a result, the advised number of servers $\tilde{m}(t) = 0$ for all t. Let $m(t)$ be the number of servers of $\mathcal{A}$ for the instance. Define $\tau := \inf\{t \mid m(t) > 0.885t^2\}$ or $\tau = \infty$ if the infimum does not exist. We distinguish two cases depending on the value of τ.

1. First, consider the case when $\tau \leq 1.225$. Fix $T = \tau$. The optimal solution to (2.1) is $m^*(t) = 0$ for $t \in [0, T]$. The value of the optimal solution is purely because of flow time and is equal to $\textsc{Opt} = \tau^2/2$.

At time $t = \tau$, algorithm $\mathcal{A}$ has at least $m(\tau) > 0.885\tau^2$ servers. The flow time is at least $\int_0^\tau q(t)\mathrm{d}t \geq \int_0^\tau \int_0^t 1 - 0.885s^2 \mathrm{d}s\mathrm{d}t \geq \tau^2/2 - 0.885\tau^4/12$ because $m(t) \leq 0.885t^2$ for $t \in [0, \tau)$. The cost of $\mathcal{A}$ is, therefore, at least $\textsc{Cost}^{\lambda}(m, T) \geq 0.885\tau^2 + \tau^2/2 - 0.885\tau^4/12 \geq 2.549 \cdot \tau^2/2 = 2.549 \cdot \textsc{Opt}$, where the second inequality follows because $\tau \leq 1.225$.

2. Next, consider the case when $\tau > 1.225$. Fix $T = 3$. The optimal solution to (2.1) is $m^*(t) = 1$ for $t \in [0, T]$. The value of the optimal solution is purely because of switching cost and is equal to $\textsc{Opt} = 1$.

At time $t = 1.225$, the queue length of $\mathcal{A}$ is at least $q(1.225) \geq \int_0^{1.225} 1 - 0.885t^2 \mathrm{d}t = 0.682$. The optimal solution starting from time $t = 1.225$ is $m(t) = 1 + q(1.225)/\sqrt{2} \geq 1.483$ for $t \in (1.225, T]$. The flow time is, therefore, at least $\int_0^T q(t)\mathrm{d}t \geq \int_0^{1.225} \int_0^s 1 - 0.885s^2 \mathrm{d}s\mathrm{d}t + q(1.225)/\sqrt{2} \geq 1.067$, again because $m(t) \leq 0.885t^2$ for $t \in [0, \tau)$. The cost of $\mathcal{A}$ is, therefore, at least $\textsc{Cost}^{\lambda}(m, T) \geq 2.549 \geq 2.549 \cdot \textsc{Opt}$.

Hence, the statement follows. □

A.6. Proof of Theorem 3

Proof of Theorem 3. Fix a finite-time horizon T, arrival rate function $\lambda(\cdot)$, and initial number of servers $m(0)$. Let $\tilde{\lambda}(\cdot)$ be the predicted arrival rate. The idea to the proof is to separate the cost into the cost of the offline component and the online component. More specifically, we claim that

$$\textsc{Cost}^{\lambda}(m, T) \leq \textsc{Cost}^{\tilde{\lambda}}(m_1, T) + (\sqrt{2\omega\beta} + \theta)T \cdot \|\Delta\lambda\|_{MAE}. \tag{A.30}$$

To see why, note that

$$\begin{aligned}
q(t) &= \int_0^t (\lambda(s) - m(s)) \mathbb{1}\{q(s) > 0 \text{ or } \lambda(s) \geq m(s)\}\mathrm{d}s \\
&\leq \int_0^t (\tilde{\lambda}(s) - m_1(s)) \mathbb{1}\{q(s) > 0 \text{ or } \lambda(s) \geq m(s)\}\mathrm{d}s \\
&\quad + \int_0^t (\Delta\lambda(s) - m_2(s)) \mathbb{1}\{q(s) > 0 \text{ or } \lambda(s) \geq m(s)\}\mathrm{d}s \\
&\leq \int_0^t (\tilde{\lambda}(s) - m_1(s)) \mathbb{1}\{q_1(s) > 0 \text{ or } \lambda(s) \geq m_1(s)\}\mathrm{d}s \\
&\quad + \int_0^t (\Delta\lambda(s) - m_2(s)) \mathbb{1}\{q_2(s) > 0 \text{ or } \lambda(s) \geq m_2(s)\}\mathrm{d}s = q_1(t) + q_2(t).
\end{aligned} \tag{A.31}$$

Therefore, the flow time of the algorithm is at most the sum of the flow time of the offline component on $\tilde{\lambda}$ and the online component on $\Delta\lambda$. Similarly, because the switching cost and the power cost are linear in the number of servers $m(\cdot)$, the cost of the algorithm is at most

$$\textsc{Cost}^{\lambda}(m, T) \leq \textsc{Cost}^{\tilde{\lambda}}(m_1, T) + \textsc{Cost}^{\Delta\lambda}(m_2, T). \tag{A.32}$$

We will further bound the cost of the online component m_2. Let $[t, t+\delta) \subseteq [0,T]$ be an arbitrary time interval for $\delta > 0$ small, and let $\Delta q(t) = \int_t^{t+\delta} \Delta\lambda(s)\mathrm{d}s$. We will bound the cost because of the $\Delta q(t)$ workload received in this time interval. The number of servers $m_2(t)$ increases by $\sqrt{\omega/(2\beta)} \cdot \Delta q(t)$ in the interval. Moreover, after a time of $\sqrt{2\beta/\omega}$, the number of servers $m_2(t)$ decreases again by $\sqrt{\omega/(2\beta)} \cdot \Delta q(t)$. Throughout $[t, t+\sqrt{2\beta/\omega})$, the queue length because of this fraction of the workload decreases linearly as $q(t+s) = \Delta q(t) - \sqrt{\omega/(2\beta)} \cdot \Delta q(t) \cdot s$ until the workload is completely handled. The cost because of waiting is, therefore,

$$\omega \cdot \int_0^{\sqrt{\frac{2\beta}{\omega}}} \left(\Delta q(t) - \sqrt{\frac{\omega}{2\beta}} \cdot \Delta q(t) \cdot s \right) \mathrm{d}s = \omega \cdot \sqrt{\frac{\beta}{2\omega}} \cdot \Delta q(t). \tag{A.33}$$

Note that because δ can be chosen arbitrarily small, the waiting cost in the interval $[t, t+\delta)$ is negligible. The switching cost is $\beta \cdot \sqrt{\omega/(2\beta)} \cdot \Delta q(t)$, and the power cost is $\theta \cdot \sqrt{2\beta/\omega} \cdot \sqrt{\omega/(2\beta)} \cdot \Delta q(t) = \theta \cdot \Delta q(t)$. The cost of the online component is, therefore,

$$\textsc{Cost}^{\Delta\lambda}(m_2, T) \le \lim_{\delta \downarrow 0} \sum_{i=0}^{\lfloor T/\delta \rfloor} \left(\sqrt{2\omega\beta} + \theta \right) \cdot \Delta q(i\delta) = \left(\sqrt{2\omega\beta} + \theta \right) T \cdot \|\Delta\lambda\|_{MAE}, \tag{A.34}$$

which proves Equation (A.30) by combining (A.32) and (A.34). Similarly, let $\Delta\lambda^*(t) = (\tilde{\lambda}(t) - \lambda(t))^+$. Then, by interchanging the actual arrival rate λ and the predicted arrival rate $\tilde{\lambda}$ in Equation (A.30), we find that

$$\textsc{Cost}^{\tilde{\lambda}}(m^*, T) \le \textsc{Cost}^{\lambda}(m^{*1}, T) + (\sqrt{2\omega\beta} + \theta) T \cdot \|\Delta\lambda\|_{MAE}, \tag{A.35}$$

where $m^*(t) = m^{*1}(t) + m^{*2}(t)$ and

$$m^{*1} \in \operatorname*{arg\,min}_{m:(0,T] \to \mathbb{R}_+} \textsc{Cost}^{\lambda}(m, T), \tag{A.36}$$

$$\frac{\mathrm{d}m^{*2}(t)}{\mathrm{d}t} = \sqrt{\frac{\omega}{2\beta}} \cdot \left(\Delta\lambda^*(t) - \Delta\lambda^*\left(t - \sqrt{2\beta/\omega} \right) \right). \tag{A.37}$$

Finally, we combine (A.30) and (A.35) to find that

$$\begin{aligned} \textsc{Cost}^{\lambda}(m, T) &\le \textsc{Cost}^{\tilde{\lambda}}(m_1, T) + \left(\sqrt{2\omega\beta} + \theta \right) T \cdot \|\Delta\lambda\|_{MAE} \\ &\le \textsc{Cost}^{\tilde{\lambda}}(m^*, T) + \left(\sqrt{2\omega\beta} + \theta \right) T \cdot \|\Delta\lambda\|_{MAE} \\ &\le \textsc{Cost}^{\lambda}(m^{*1}, T) + \left(\sqrt{2\omega\beta} + \theta \right) T \cdot \|\tilde{\lambda} - \lambda\|_{MAE}, \end{aligned} \tag{A.38}$$

where the first inequality follows by (A.30), the second inequality follows because m_1 achieves the minimum cost on $\tilde{\lambda}$, and the third inequality follows by (A.35). This completes the proof because m^{*1} is the optimal offline solution on λ. □

A.7. Proof of Proposition 3

Proof of Proposition 3. Fix any algorithm $\mathcal{A}$, and let $\mathrm{CR}(\eta)$ denote its competitive ratio when it has access to an η-accurate prediction. Fix $\delta > 0$, and assume that $\mathrm{CR}(0) \le 1 + \delta$. We will construct an instance for which $\textsc{Cost}^{\lambda}(m, T) \ge \textsc{Opt}/(4\delta)$.

Let $T = 2 + \sqrt{2}\delta$ and $\lambda(t) = 2$ for $t \in [0, \sqrt{2}\delta)$. The value of $\lambda(t)$ for $t \in [\sqrt{2}\delta, T]$ will be specified later. Fix $\beta = \omega = 1$ and $\theta = 0$. Let the prediction $\tilde{\lambda}(t) = 2$ for $t \in [0,T]$, and let $m(t)$ be the number of servers of $\mathcal{A}$ for the instance. We distinguish two cases depending on the value of $\overline{m} := \sup_{t \in [0, \sqrt{2}\delta)} m(t)$.

1. First, consider the case when $\overline{m} < 1$. Fix $\lambda(t) = 2$ for $t \in [\sqrt{2}\delta, T]$. One feasible solution of (2.1) is $m^*(t) = 2$ for $t \in (0,T]$. The cost of this solution is purely because of switching cost and therefore, $\textsc{Opt} \le 2$. As $\tilde{\lambda}(t) = \lambda(t)$, the prediction $\tilde{\lambda}$ is perfect. Hence, by the assumption that $\mathcal{A}$ is $(1+\delta)$ consistent, we must have $\textsc{Cost}^{\lambda}(m, T) \le (1+\delta) \cdot \textsc{Opt}$ for this instance. We will see that this cannot be achieved under the case $\overline{m} < 1$.

Note at time $t = \sqrt{2}\delta$, the queue length of $\mathcal{A}$ is at least $q(\sqrt{2}\delta) \ge \int_0^{\sqrt{2}\delta} (2 - \overline{m})\mathrm{d}t > \sqrt{2}\delta$. The optimal solution starting from time $t = \sqrt{2}\delta$ is $m(t) = 2 + q(\sqrt{2}\delta)/\sqrt{2} > 4 + \delta$ for $t \in (\sqrt{2}\delta, T]$. As a result, the flow time is at least $\int_0^T q(t)\mathrm{d}t \ge q(\sqrt{2}\delta)/\sqrt{2} > \delta$. The cost of $\mathcal{A}$ is, therefore, at least $\textsc{Cost}^{\lambda}(m, T) > 2 + 2\delta \ge (1+\delta) \cdot \textsc{Opt}$. This is a contradiction with our assumption that $\mathcal{A}$ is $(1+\delta)$ consistent, and hence, the next case must occur.

2. Next, consider the case when $\overline{m} \ge 1$. Fix $\lambda(t) = 0$ for $t \in [\sqrt{2}\delta, T]$. One feasible solution of (2.1) is $m^*(t) = 2\delta$ for $t \in [0,T]$. The cost of the optimal solution is, therefore, at most $\textsc{Opt} \le 4\delta - 2\delta^2 \le 4\delta$. Now, the cost of $\mathcal{A}$ is at least $\textsc{Cost}^{\lambda}(m, T) \ge 1 \ge \textsc{Opt}/(4\delta)$ because of switching cost. Moreover, the MAE is $T \cdot \|\tilde{\lambda} - \lambda\|_{MAE} = \int_0^T |\tilde{\lambda}(t) - \lambda(t)| \mathrm{d}t = 4$, and hence, the prediction is $1/\delta$ accurate.

Hence, Equation (4.17) follows. □

A.8. Proof of Proposition 4

Proof of Proposition 4. Fix any $\omega, \beta, \theta > 0$ and any function $\tau : \mathbb{R}_+^3 \to (0, \infty)$. Let $m(t)$ be the number of servers of the timer algorithm for the current instance. We will construct a sequence of instances for which $\textsc{Cost}^{\lambda}(m, T)/\textsc{Opt} \to \infty$.

Let $\tau = \tau(\omega, \beta, \theta)$, and fix $0 < \varepsilon < \tau$. Let $\lambda(t) = 2$ for $t \in [0, t_0]$ and $\lambda(t_0 + i\tau + s) = \mathbb{1}\{s \in (0, \varepsilon]\}$ for $s \in (0, \tau]$ and $i \in \mathbb{N} \cap [0, T]$, where $t_0 = \inf\{t \in \mathbb{R} \mid m(t) \geq 1\}$ or $t_0 = \infty$ if the infimum does not exist. We let T be sufficiently large. Let us distinguish two cases depending on the value of t_0.

1. First, consider the case when $t_0 = \infty$. One feasible solution of (2.1) is $m^*(t) = 2$. This solution does not incur any waiting cost, and the cost of the optimal solution is, therefore, at most $\textsc{Opt} \leq 2\beta + \theta T$. However, because $m(t) < 1$, the cost of the timer algorithm is at least $\textsc{Cost}^\lambda(m, T) = \omega \cdot \int_0^T s \mathrm{d}s = \frac{\omega T^2}{2}$. Then, $\textsc{Cost}^\lambda(m, T)/\textsc{Opt} \geq \frac{\omega T^2}{4\beta + 2\theta T} \to \infty$ as $T \to \infty$.

2. Next, consider the case when $t_0 < \infty$. One possible solution of (2.1) is $m^*(t) = 2$ for $t \in [0, t_0]$ and $m^*(t) = \varepsilon/\tau$ for $t \in (t_0, T]$. The cost of the optimal solution is, therefore, at most $\textsc{Opt} \leq 2\beta + 2\theta t_0 + \varepsilon\theta T/\tau + \omega T/\tau \cdot \int_0^\varepsilon (1 - \varepsilon/\tau) s \mathrm{d}s + \omega T/\tau \cdot \int_\varepsilon^\tau (\varepsilon - \varepsilon s/\tau) \mathrm{d}s \leq 2\beta + 2\theta t_0 + \varepsilon\theta T/\tau + \varepsilon\omega T/2$. However, note that after time $t \geq t_0$, a server idles for at most $\tau - \varepsilon < \tau$ time, which means that the timer algorithm maintains at least one server throughout $[t_0, T]$. Therefore, the cost of the timer algorithm is at least $\textsc{Cost}^\lambda(m, T) = \beta + \theta(T - t_0)$. Then, $\textsc{Cost}^\lambda(m, T)/\textsc{Opt} \geq \frac{\beta + \theta(T - t_0)}{2\beta + 2\theta t_0 + \varepsilon\theta T/\tau + \varepsilon\omega T/2} \to \infty$ as $T \to \infty$ and $\varepsilon \to 0$. □

A.9. Proof of Proposition 5

Proof of Proposition 5. Fix a finite-time horizon T, arrival rate function $\lambda(\cdot)$, and initial number of servers $m(0)$. Let $m^*(t)$ be a solution of the offline optimization Problem (2.1) and $q^*(t)$ the corresponding workload. Assume that the solution achieves a finite cost. If there does not exist a solution that achieves finite cost, then Proposition 5 follows immediately. Without loss of generality, assume that $m^*(t)$ is differentiable. To see why this is possible, assume that $m^*(t)$ is not differentiable. Define the interpolation $m^*_\delta(t)$ of $m^*(t)$ such that

$$m^*_\delta(t) = \int_t^{t+\delta} \frac{m^*(s)}{\delta} \mathrm{d}s, \tag{A.39}$$

which is differentiable for all $\delta > 0$. Also, note that

$$\int_{t_1}^{t_2} m^*_\delta(t) \mathrm{d}t = \int_{t_1}^{t_2} \int_t^{t+\delta} \frac{m^*(s)}{\delta} \mathrm{d}s \, \mathrm{d}t \to \int_{t_1}^{t_2} m^*(t) \, \mathrm{d}t \text{ as } \delta \to 0 \tag{A.40}$$

for any $0 \leq t_1 \leq t_2 \leq \infty$. The cost of $m^*(t)$ and $m^*_\delta(t)$, therefore, coincides asymptotically as $\delta \to 0$. As a result, each function $m^*(\cdot)$ can be written as the limit of a sequence of differentiable functions $m^*_\delta(\cdot)$, and we, therefore, assume that $m^*(t)$ is differentiable without loss of generality.

A.9.1. Overview of the Proof. Let $m(t)$ be the number of servers of ABCS (Algorithm 3) and $q(t)$ be the corresponding workload. The goal of the proof will be to find a nonnegative potential function $\Phi(t)$ such that

$$\frac{\mathrm{d}\Phi(t)}{\mathrm{d}t} + \frac{\partial \textsc{Cost}(m, t)}{\partial t} \leq \mathrm{PCR} \cdot \frac{\partial \textsc{Cost}(m^*, t)}{\partial t}, \tag{A.41}$$

where we have omitted λ from the notation $\textsc{Cost}^\lambda(m, t)$. Note that $\textsc{Cost}(m, t)$ and $\textsc{Cost}(m^*, t)$ are differentiable because $m(t)$ and $m^*(t)$ are differentiable. We integrate Equation (A.41) from time $t = 0$ to $t = T$ to obtain

$$\textsc{Cost}(m, T) \leq \mathrm{PCR} \cdot \textsc{Cost}(m^*, T) + \Phi(0) - \Phi(T) \leq \mathrm{PCR} \cdot \textsc{Cost}(m^*, T) + \Phi(0), \tag{A.42}$$

where the last step follows because $\Phi(T)$ is nonnegative. The proof of Proposition 5 is, therefore, completed if we manage to find a differentiable potential function $\Phi(t)$ satisfying Equation (A.41) and $\Phi(0) = \frac{\beta \cdot m(0)}{r_2} = 0$. If $m(0) > 0$ instead, then a similar statement as in Proposition 5 holds but with an additive term of $\frac{\beta \cdot m(0)}{r_2}$.

A.9.2. Choice of $\Phi(t)$. Define the potential function $\Phi(t)$ such that

$$\Phi(t) = \begin{cases} c_5\beta \cdot (d_{R_1}(t) - m(t) + m^*(t)), & \text{if } m(t) > m^*(t) \\ c_6\beta \cdot (d_{r_1}(t) - m(t) + m^*(t)) & \text{if } m(t) \leq m^*(t) \end{cases}$$
$$+ \frac{\beta \cdot m(t)}{r_2} + c_6 R_2 \theta \cdot [q(t) - q^*(t)]^+, \tag{A.43}$$

where

$$d_r(t) = \sqrt{\frac{r\omega \cdot ([q(t) - q^*(t)]^+)^2}{\beta} + (m(t) - m^*(t))^2}. \tag{A.44}$$

Note that $\Phi(t)$ is nonnegative and $\Phi(0) = \frac{\beta \cdot m(0)}{r_2}$. The sophisticated reader might remark that there are points in the domain for which $\Phi(t)$ is not differentiable. As there can only be countably many of these points, these points do not influence the integral of Equation (A.41), and we simply ignore these points in the analysis.

A.9.3. Verification of (A.41). We continue by verifying Equation (A.41). We distinguish four cases, depending on whether $q(t) > q^*(t)$ or $q(t) \leq q^*(t)$ and $m(t) > m^*(t)$ or $m(t) \leq m^*(t)$.

i. Assume that $q(t) > q^*(t)$ and $m(t) > m^*(t)$. Recall that by definition,

$$\frac{\mathrm{d}q}{\mathrm{d}t} = \lambda(t) - m(t), \frac{\mathrm{d}m}{\mathrm{d}t} = \frac{\hat{r}_1(t)\omega \cdot q(t) - \hat{r}_2(t)\theta \cdot m(t)}{\beta} \leq \frac{R_1\omega \cdot q(t)}{\beta}. \tag{A.45}$$

The derivative of $d_{R_1}(t)$ is, therefore, at most

$$\begin{aligned}\beta \cdot \frac{\mathrm{d}d_{R_1}(t)}{\mathrm{d}t} &\leq d_{R_1}(t)^{-1} \cdot \begin{pmatrix} R_1\omega \cdot (q(t) - q^*(t))(\lambda(t) - m(t)) \\ +R_1\omega \cdot (q^*(t) - q(t))(\lambda(t) - m^*(t)) \\ +R_1\omega \cdot q(t) \cdot (m(t) - m^*(t)) \\ +\beta \cdot \frac{\mathrm{d}m^*}{\mathrm{d}t} \cdot (m^*(t) - m(t)) \end{pmatrix} \\ &= d_{R_1}(t)^{-1} \cdot (m(t) - m^*(t)) \left(R_1\omega \cdot q^*(t) - \beta \cdot \frac{\mathrm{d}m^*}{\mathrm{d}t} \right) \\ &\leq R_1\omega \cdot q^*(t) + \beta \cdot \left[-\frac{\mathrm{d}m^*}{\mathrm{d}t} \right]^+. \end{aligned} \tag{A.46}$$

The derivative of the potential function $\Phi(t)$ is then

$$\begin{aligned}\frac{\mathrm{d}\Phi(t)}{\mathrm{d}t} \leq\ & c_5R_1\omega \cdot q^*(t) + c_5\beta \cdot \left(\left[-\frac{\mathrm{d}m^*}{\mathrm{d}t} \right]^+ + \frac{\mathrm{d}m^*}{\mathrm{d}t} \right) - \left(c_5 - \frac{1}{r_2} \right) \beta \cdot \frac{\mathrm{d}m}{\mathrm{d}t} \\ &+\ c_6R_2\theta \cdot (\lambda(t) - m(t) - \lambda(t) + m^*(t)) \\ \leq\ & c_5R_1\omega \cdot q^*(t) + c_5\beta \cdot \left[\frac{\mathrm{d}m^*}{\mathrm{d}t} \right]^+ - \left(1 + \frac{1}{r_1} \right) \beta \cdot \frac{\mathrm{d}m}{\mathrm{d}t} \\ &+ \left(1 + R_2 + \frac{R_2}{r_1} \right) \theta \cdot (m^*(t) - m(t)). \end{aligned} \tag{A.47}$$

The derivative of the cumulative cost $\textsc{Cost}(m,t)$ is

$$\begin{aligned}\frac{\partial \textsc{Cost}(m,t)}{\partial t} &= \omega \cdot q(t) + \beta \cdot \left[\frac{\mathrm{d}m}{\mathrm{d}t} \right]^+ + \theta \cdot m(t) \\ &\leq \frac{\beta}{r_1} \cdot \frac{\mathrm{d}m}{\mathrm{d}t} + \beta \cdot \left[\frac{\mathrm{d}m}{\mathrm{d}t} \right]^+ + \left(1 + \frac{R_2}{r_1} \right) \theta \cdot m(t). \end{aligned} \tag{A.48}$$

We sum Equations (A.47) and (A.48) and cancel terms to obtain

$$\begin{aligned}\frac{\mathrm{d}\Phi(t)}{\mathrm{d}t} + \frac{\partial \textsc{Cost}(m,t)}{\partial t} &\leq c_5R_1\omega \cdot q^*(t) + c_5\beta \cdot \left[\frac{\mathrm{d}m^*}{\mathrm{d}t} \right]^+ + \left(1 + R_2 + \frac{R_2}{r_1} \right) \theta \cdot m^*(t) \\ &\leq \mathrm{PCR} \cdot \frac{\partial \textsc{Cost}(m^*,t)}{\partial t}. \end{aligned} \tag{A.49}$$

Note that if $\frac{\mathrm{d}m}{\mathrm{d}t} \geq 0$, then the sum follows immediately. If $\frac{\mathrm{d}m}{\mathrm{d}t} < 0$, we apply the bound

$$-\beta \cdot \frac{\mathrm{d}m}{\mathrm{d}t} \leq R_2\theta \cdot m(t) - r_1\omega \cdot q(t) \leq R_2\theta \cdot m(t). \tag{A.50}$$

ii. Assume that $q(t) \leq q^*(t)$ and $m(t) > m^*(t)$. The potential function $\Phi(t)$ simplifies to

$$\Phi(t) = \frac{\beta \cdot m(t)}{r_2}. \tag{A.51}$$

The derivative of the potential function $\Phi(t)$ is then

$$\frac{\mathrm{d}\Phi(t)}{\mathrm{d}t} = \frac{\beta}{r_2} \cdot \frac{\mathrm{d}m}{\mathrm{d}t} \leq \frac{R_1\omega}{r_2} \cdot q^*(t) - \theta \cdot m(t). \tag{A.52}$$

The derivative of the cumulative cost $\textsc{Cost}(m,t)$ is

$$\begin{aligned}\frac{\partial \textsc{Cost}(m,t)}{\partial t} &= \omega \cdot q(t) + \beta \cdot \left[\frac{\mathrm{d}m}{\mathrm{d}t} \right]^+ + \theta \cdot m(t) \\ &\leq \omega \cdot q(t) + \beta \cdot [R_1\omega \cdot q(t) - r_2\theta \cdot m(t)]^+ + \theta \cdot m(t) \\ &\leq (1 + R_1)\omega \cdot q^*(t) + \theta \cdot m(t) \end{aligned} \tag{A.53}$$

.

We sum Equations (A.52) and (A.53) and cancel terms to obtain

$$\frac{\mathrm{d}\Phi(t)}{\mathrm{d}t}+\frac{\partial \textsc{Cost}(m,t)}{\partial t}\le\left(1+R_1+\frac{R_1}{r_2}\right)\omega\cdot q^*(t)\le \mathrm{PCR}\cdot\frac{\partial \textsc{Cost}(m^*,t)}{\partial t}. \tag{A.54}$$

iii. Assume that $q(t)>q^*(t)$ and $m(t)\le m^*(t)$. Recall that, by definition,

$$\frac{\mathrm{d}q}{\mathrm{d}t}=\lambda(t)-m(t),\ \frac{\mathrm{d}m}{\mathrm{d}t}=\frac{\hat{r}_1(t)\omega\cdot q(t)-\hat{r}_2(t)\theta\cdot m(t)}{\beta}\ge\frac{r_1\omega\cdot q(t)-R_2\theta\cdot m(t)}{\beta}. \tag{A.55}$$

The derivative of $d_{r_1}(t)$ is, therefore, at most

$$\begin{aligned}\beta\cdot\frac{\mathrm{d}d_{r_1}(t)}{\mathrm{d}t}&\le d_{r_1}(t)^{-1}\cdot\begin{pmatrix}r_1\omega\cdot(q(t)-q^*(t))(\lambda(t)-m(t))\\+r_1\omega\cdot(q^*(t)-q(t))(\lambda(t)-m^*(t))\\+(r_1\omega\cdot q(t)-R_2\theta\cdot m(t))(m(t)-m^*(t))\\+\beta\cdot\dfrac{\mathrm{d}m^*}{\mathrm{d}t}\cdot(m^*(t)-m(t))\end{pmatrix}\\&\le d_{r_1}(t)^{-1}\cdot(m^*(t)-m(t))\left(R_2\theta\cdot m(t)+\beta\cdot\frac{\mathrm{d}m^*}{\mathrm{d}t}\right)\\&\le R_2\theta\cdot m(t)+\beta\cdot\left[\frac{\mathrm{d}m^*}{\mathrm{d}t}\right]^+.\end{aligned} \tag{A.56}$$

The derivative of the potential function $\Phi(t)$ is then

$$\begin{aligned}\frac{\mathrm{d}\Phi(t)}{\mathrm{d}t}&\le c_6R_2\theta\cdot m(t)+c_6\beta\cdot\left(\left[\frac{\mathrm{d}m^*}{\mathrm{d}t}\right]^+-\frac{\mathrm{d}m^*}{\mathrm{d}t}\right)-\left(c_6-\frac{1}{r_2}\right)\beta\cdot\frac{\mathrm{d}m}{\mathrm{d}t}\\&\quad+c_6R_2\theta\cdot(\lambda(t)-m(t)-\lambda(t)+m^*(t))\\&\le c_6R_2\theta\cdot m^*(t)+2c_6\beta\cdot\left[\frac{\mathrm{d}m^*}{\mathrm{d}t}\right]^+-\left(c_6-\frac{1}{r_2}\right)\beta\cdot\frac{\mathrm{d}m}{\mathrm{d}t}.\end{aligned} \tag{A.57}$$

Similar to before, the derivative of the cumulative cost $\textsc{Cost}(m,t)$ is

$$\frac{\partial \textsc{Cost}(m,t)}{\partial t}\le\frac{\beta}{r_1}\cdot\frac{\mathrm{d}m}{\mathrm{d}t}+\beta\cdot\left[\frac{\mathrm{d}m}{\mathrm{d}t}\right]^+-\left(1+\frac{R_2}{r_1}\right)\theta\cdot m(t). \tag{A.58}$$

We sum Equations (A.57) and (A.58) and cancel terms to obtain

$$\begin{aligned}\frac{\mathrm{d}\Phi(t)}{\mathrm{d}t}+\frac{\partial \textsc{Cost}(m,t)}{\partial t}&\le 2c_6\beta\cdot\left[\frac{\mathrm{d}m^*}{\mathrm{d}t}\right]^+-\left(2c_6R_2+1-\frac{R_2}{r_2}\right)\theta\cdot m^*(t)\\&\le \mathrm{PCR}\cdot\frac{\partial \textsc{Cost}(m^*,t)}{\partial t}.\end{aligned} \tag{A.59}$$

iv. Assume that $q(t)\le q^*(t)$ and $m(t)\le m^*(t)$. The potential function $\Phi(t)$ simplifies to

$$\Phi(t)=2c_6\beta\cdot(m^*(t)-m(t))+\frac{\beta\cdot m(t)}{r_2}. \tag{A.60}$$

The derivative of the potential function $\Phi(t)$ is then

$$\frac{\mathrm{d}\Phi(t)}{\mathrm{d}t}=2c_6\beta\cdot\frac{\mathrm{d}m^*}{\mathrm{d}t}-\left(2c_6-\frac{1}{r_2}\right)\beta\cdot\frac{\mathrm{d}m}{\mathrm{d}t}. \tag{A.61}$$

Similar to before, the derivative of the cumulative cost $\textsc{Cost}(m,t)$ is

$$\frac{\partial \textsc{Cost}(m,t)}{\partial t}\le\frac{\beta}{r_1}\cdot\frac{\mathrm{d}m}{\mathrm{d}t}+\beta\cdot\left[\frac{\mathrm{d}m}{\mathrm{d}t}\right]^+-\left(1+\frac{R_2}{r_1}\right)\theta\cdot m(t). \tag{A.62}$$

We sum Equations (A.61) and (A.62) and cancel terms to obtain

$$\begin{aligned}\frac{\mathrm{d}\Phi(t)}{\mathrm{d}t}+\frac{\partial \textsc{Cost}(m,t)}{\partial t}&\le 2c_6\beta\cdot\left[\frac{\mathrm{d}m^*}{\mathrm{d}t}\right]^+-\left(2c_6R_2+1-\frac{R_2}{r_2}\right)\theta\cdot m^*(t)\\&\le \mathrm{PCR}\cdot\frac{\partial \textsc{Cost}(m^*,t)}{\partial t}.\end{aligned} \tag{A.63}$$

□

A.10. Proof of Proposition 6

Proof of Proposition 6. Fix a finite-time horizon T, arrival rate function $\lambda(\cdot)$, and initial number of servers $m(0)$. Let $\tilde{m}(\cdot)$ be the number of advised servers of AP and $\tilde{q}(\cdot)$ be the corresponding workload. Assume that the advised number of

servers achieves finite cost. If the advised number of servers does not achieve finite cost, then Proposition 6 follows immediately. Without loss of generality, similar to the proof of Proposition 5, assume that $\tilde{m}(t)$ is differentiable.

A.10.1. Overview of the Proof. As argued before (see the proof of Proposition 5), the proof of Proposition 6 requires us to find a nonnegative potential function $\Phi(t)$ such that

$$\frac{\mathrm{d}\Phi(t)}{\mathrm{d}t} + \frac{\partial \textsc{Cost}(m,t)}{\partial t} \leq \mathrm{OCR} \cdot \frac{\partial \textsc{Cost}(\tilde{m},t)}{\partial t}, \tag{A.64}$$

where we have omitted λ from the notation $\textsc{Cost}^{\lambda}(m,t)$.

A.10.2. Choice of $\Phi(t)$. Define the potential function $\Phi(t)$ such that

$$\begin{aligned}\Phi(t) = &\begin{cases} c_1\beta \cdot (d_{r_1}(t) - m(t) + \tilde{m}(t)) & \text{if } \hat{r}_1(t) = r_1, \\ c_2\beta \cdot d_{R_1}(t) - c_3\beta \cdot (m(t) - \tilde{m}(t)) & \text{if } \hat{r}_1(t) = R_1, \end{cases} \\ &+ \frac{\beta \cdot m(t)}{R_2} + c_4\theta \cdot [q(t) - \tilde{q}(t)]^+, \end{aligned} \tag{A.65}$$

where

$$d_r(t) = \sqrt{\frac{r\omega \cdot ([q(t) - \tilde{q}(t)]^+)^2}{\beta} + (m(t) - \tilde{m}(t))^2}. \tag{A.66}$$

Note that $\Phi(0) = \frac{\beta \cdot m(0)}{R_2} = 0$. If $m(0) > 0$ instead, then a similar statement as in Proposition 6 holds but with an additive term of $\frac{\beta \cdot m(0)}{R_2}$. If $\hat{r}_1(t) = r_1$ or $m(t) \leq \tilde{m}(t)$, then $\Phi(t)$ is trivially nonnegative. Assume that $\hat{r}_1(t) = R_1$ and $m(t) > \tilde{m}(t)$. Then,

$$\Phi(t) \geq c_2\beta \cdot d_{R_1}(t) - c_3\beta \cdot (m(t) - \tilde{m}(t)) \geq (c_2\sqrt{1 + 2R_1} - c_3)\beta \cdot (m(t) - \tilde{m}(t)) \geq 0, \tag{A.67}$$

and hence, $\Phi(t)$ is nonnegative. The sophisticated reader might remark that there are points in the domain for which $\Phi(t)$ is not differentiable. As there can only be countably many of these points, these points do not influence the integral of Equation (A.64), and we simply ignore these points in the analysis.

A.10.3. Verification of (A.64). We continue by verifying Equation (A.64). We distinguish eight cases depending on whether $q(t) > \tilde{q}(t)$ or $q(t) \leq \tilde{q}(t)$, $m(t) > \tilde{m}(t)$ or $m(t) \leq \tilde{m}(t)$, and $\hat{r}_1(t) = r_1$ or $\hat{r}_1(t) = R_1$.

i, a. Assume that $q(t) > \tilde{q}(t)$, $m(t) > \tilde{m}(t)$, and $\hat{r}_1(t) = r_1$. Note that $\hat{r}_2(t) = r_2$ because $q(t) > \tilde{q}(t)$. Recall that by definition,

$$\frac{\mathrm{d}q(t)}{\mathrm{d}t} = \lambda(t) - m(t), \frac{\mathrm{d}m(t)}{\mathrm{d}t} = \frac{r_1\omega \cdot q(t) - r_2\theta \cdot m(t)}{\beta}. \tag{A.68}$$

The derivative of $d_{r_1}(t)$ is, therefore, at most

$$\begin{aligned}\beta \cdot \frac{\mathrm{d}d_{r_1}(t)}{\mathrm{d}t} &\leq d_{r_1}(t)^{-1} \cdot \begin{pmatrix} r_1\omega \cdot (q(t) - \tilde{q}(t))(\lambda(t) - m(t)) \\ + r_1\omega \cdot (\tilde{q}(t) - q(t))(\lambda(t) - \tilde{m}(t)) \\ + (r_1\omega \cdot q(t) - r_2\theta \cdot m(t))(m(t) - \tilde{m}(t)) \\ + \beta \cdot \frac{\mathrm{d}\tilde{m}}{\mathrm{d}t} \cdot (\tilde{m}(t) - m(t)) \end{pmatrix} \\ &= d_{r_1}(t)^{-1} \cdot (m(t) - \tilde{m}(t)) \left(r_1\omega \cdot \tilde{q}(t) - r_2\theta \cdot m(t) - \beta \cdot \frac{\mathrm{d}\tilde{m}}{\mathrm{d}t} \right) \\ &\leq r_1\omega \cdot \tilde{q}(t) - \frac{r_2\theta}{\sqrt{1 + 2r_1}} \cdot m(t) + \beta \cdot \left[-\frac{\mathrm{d}\tilde{m}}{\mathrm{d}t} \right]^+ - \frac{\beta}{\sqrt{1 + 2r_1}} \cdot \left[\frac{\mathrm{d}\tilde{m}}{\mathrm{d}t} \right]^+. \end{aligned} \tag{A.69}$$

The derivative of the potential function $\Phi(t)$ is then

$$\begin{aligned}\frac{\mathrm{d}\Phi(t)}{\mathrm{d}t} &\leq c_1 r_1\omega \cdot \tilde{q}(t) - \frac{c_1 r_2\theta}{\sqrt{1 + 2r_1}} \cdot m(t) \\ &\quad + c_1\beta \cdot \left[-\frac{\mathrm{d}\tilde{m}}{\mathrm{d}t} \right]^+ - \frac{c_1\beta}{\sqrt{1 + 2r_1}} \cdot \left[\frac{\mathrm{d}\tilde{m}}{\mathrm{d}t} \right]^+ + c_1\beta \cdot \frac{\mathrm{d}\tilde{m}}{\mathrm{d}t} \\ &\quad - \left(c_1 - \frac{1}{R_2} \right) \beta \cdot \frac{\mathrm{d}m}{\mathrm{d}t} + c_4\theta \cdot (\lambda(t) - m(t) - \lambda(t) + \tilde{m}(t)) \\ &\leq c_1 r_1\omega \cdot \tilde{q}(t) - \frac{r_2\theta}{r_1} \cdot m(t) + \left(1 + \frac{1}{R_2} \right) \beta \cdot \left[\frac{\mathrm{d}\tilde{m}}{\mathrm{d}t} \right]^+ \\ &\quad - \left(1 + \frac{1}{r_1} \right) \beta \cdot \frac{\mathrm{d}m}{\mathrm{d}t} + c_4\theta \cdot (\tilde{m}(t) - m(t)). \end{aligned} \tag{A.70}$$

The derivative of the cumulative cost $\textsc{Cost}(m,t)$ is

$$\begin{aligned}\frac{\partial \textsc{Cost}(m,t)}{\partial t} &= \omega \cdot q(t) + \beta \cdot \left[\frac{\mathrm{d}m}{\mathrm{d}t}\right]^+ + \theta \cdot m(t) \\ &= \frac{\beta}{r_1} \cdot \frac{\mathrm{d}m}{\mathrm{d}t} + \beta \cdot \left[\frac{\mathrm{d}m}{\mathrm{d}t}\right]^+ + \left(1 + \frac{r_2}{r_1}\right)\theta \cdot m(t). \end{aligned} \tag{A.71}$$

We sum Equations (A.70) and (A.71) and cancel terms to obtain

$$\begin{aligned}\frac{\mathrm{d}\Phi(t)}{\mathrm{d}t} + \frac{\partial \textsc{Cost}(m,t)}{\partial t} &\le c_1 r_1 \omega \cdot \tilde{q}(t) + \left(1 + \frac{1}{R_2}\right)\beta \cdot \left[\frac{\mathrm{d}\tilde{m}}{\mathrm{d}t}\right]^+ + c_4 \theta \cdot \tilde{m}(t) \\ &\le \mathrm{OCR} \cdot \frac{\partial \textsc{Cost}(\tilde{m},t)}{\partial t}. \end{aligned} \tag{A.72}$$

Note that if $\frac{\mathrm{d}m(t)}{\mathrm{d}t} \ge 0$, then the sum follows immediately. If $\frac{\mathrm{d}m(t)}{\mathrm{d}t} < 0$, we apply the bound

$$-\beta \cdot \frac{\mathrm{d}m(t)}{\mathrm{d}t} = r_2 \theta \cdot m(t) - r_1 \omega \cdot q(t) \le r_2 \theta \cdot m(t). \tag{A.73}$$

i, b. Assume that $q(t) > \tilde{q}(t)$, $m(t) > \tilde{m}(t)$, and $\hat{r}_1(t) = R_1$. Note that $\hat{r}_2(t) = r_2$ because $q(t) > \tilde{q}(t)$. The derivative of $d_{R_1}(t)$ is, therefore, at most

$$\begin{aligned}\beta \cdot \frac{\mathrm{d}d_{R_1}(t)}{\mathrm{d}t} &\le d_{R_1}(t)^{-1} \cdot \begin{pmatrix} R_1\omega \cdot (q(t) - \tilde{q}(t))(\lambda(t) - m(t)) \\ + R_1\omega \cdot (\tilde{q}(t) - q(t))(\lambda(t) - \tilde{m}(t)) \\ + R_1\omega \cdot q(t) \cdot (m(t) - \tilde{m}(t)) \\ + \beta \cdot \frac{\mathrm{d}\tilde{m}}{\mathrm{d}t} \cdot (\tilde{m}(t) - m(t)) \end{pmatrix} \\ &= d_{R_1}(t)^{-1} \cdot (m(t) - \tilde{m}(t)) \left(R_1\omega \cdot \tilde{q}(t) - \beta \cdot \frac{\mathrm{d}\tilde{m}}{\mathrm{d}t}\right) \\ &\le \frac{R_1\omega}{\sqrt{1 + 2R_1}} \cdot \tilde{q}(t) + \frac{\beta}{\sqrt{1 + 2R_1}} \cdot \left[-\frac{\mathrm{d}\tilde{m}}{\mathrm{d}t}\right]^+. \end{aligned} \tag{A.74}$$

The derivative of the potential function $\Phi(t)$ is then

$$\begin{aligned}\frac{\mathrm{d}\Phi(t)}{\mathrm{d}t} &\le \frac{c_2 R_1 \omega}{\sqrt{1 + 2R_1}} \cdot \tilde{q}(t) + \frac{c_2 \beta}{\sqrt{1 + 2R_1}} \cdot \left[-\frac{\mathrm{d}\tilde{m}}{\mathrm{d}t}\right]^+ + c_3 \beta \cdot \frac{\mathrm{d}\tilde{m}}{\mathrm{d}t} \\ &\quad - \left(c_3 - \frac{1}{R_2}\right)\frac{\mathrm{d}m}{\mathrm{d}t} + c_4 \theta \cdot (\lambda(t) - m(t) - \lambda(t) + \tilde{m}(t)) \\ &\le \frac{c_2 R_1 \omega}{\sqrt{1 + 2R_1}} \cdot \tilde{q}(t) + c_3 \beta \cdot \left[\frac{\mathrm{d}\tilde{m}}{\mathrm{d}t}\right]^+ \\ &\quad - \left(1 + \frac{1}{R_1}\right)\beta \cdot \frac{\mathrm{d}m}{\mathrm{d}t} + c_4 \theta \cdot (\tilde{m}(t) - m(t)). \end{aligned} \tag{A.75}$$

The derivative of the cumulative cost $\textsc{Cost}(m,t)$ is

$$\frac{\partial \textsc{Cost}(m,t)}{\partial t} = \frac{\beta}{R_1} \cdot \frac{\mathrm{d}m}{\mathrm{d}t} + \beta \cdot \left[\frac{\mathrm{d}m}{\mathrm{d}t}\right]^+ + \left(1 + \frac{r_2}{R_1}\right)\theta \cdot m(t). \tag{A.76}$$

We sum Equations (A.75) and (A.76) and cancel terms to obtain

$$\begin{aligned}\frac{\mathrm{d}\Phi(t)}{\mathrm{d}t} + \frac{\partial \textsc{Cost}(m,t)}{\partial t} &\le \frac{c_2 R_1}{\sqrt{1 + 2R_1}} \omega \cdot \tilde{q}(t) + c_3 \beta \cdot \left[\frac{\mathrm{d}\tilde{m}}{\mathrm{d}t}\right]^+ + c_4 \theta \cdot \tilde{m}(t) \\ &\le \mathrm{OCR} \cdot \frac{\partial \textsc{Cost}(\tilde{m},t)}{\partial t}. \end{aligned} \tag{A.77}$$

ii, a. Assume that $q(t) \le \tilde{q}(t)$, $m(t) > \tilde{m}(t)$, and $\hat{r}_1(t) = r_1$. Note that $\hat{r}_2(t) = R_2$ because $m(t) > \tilde{m}(t)$ and $q(t) \le \tilde{q}(t)$. The potential function $\Phi(t)$ simplifies to

$$\Phi(t) = \frac{\beta \cdot m(t)}{R_2}. \tag{A.78}$$

The derivative of the potential function $\Phi(t)$ is then

$$\frac{\mathrm{d}\Phi(t)}{\mathrm{d}t} = \frac{\beta}{R_2} \cdot \frac{\mathrm{d}m}{\mathrm{d}t} = \frac{r_1\omega}{R_2} \cdot q(t) - \theta \cdot m(t). \tag{A.79}$$

The derivative of the cumulative cost $\textsc{Cost}(m,t)$ is

$$\begin{aligned}\frac{\partial \textsc{Cost}(m,t)}{\partial t} &= \omega \cdot q(t) + \beta \cdot \left[\frac{\mathrm{d}m}{\mathrm{d}t}\right] + \theta \cdot m(t) \\ &= \omega \cdot q(t) + \beta \cdot \left[\frac{r_1\omega \cdot q(t)}{\beta} - \frac{R_2\theta \cdot m(t)}{\beta}\right]^+ + \theta \cdot m(t) \\ &\le (1 + r_1)\omega \cdot q(t) + \theta \cdot m(t).\end{aligned} \tag{A.80}$$

We sum Equations (A.79) and (A.80) and cancel terms to obtain

$$\frac{\mathrm{d}\Phi(t)}{\mathrm{d}t} + \frac{\partial \textsc{Cost}(m,t)}{\partial t} \le \left(1 + r_1 + \frac{r_1}{R_2}\right)\omega \cdot \tilde{q}(t) \le \mathrm{OCR} \cdot \frac{\partial \textsc{Cost}(\tilde{m},t)}{\partial t}. \tag{A.81}$$

ii, b. Assume that $q(t) \le \tilde{q}(t)$, $m(t) > \tilde{m}(t)$, and $\hat{r}_1(t) = R_1$. However, $\hat{r}_1(t) = R_1$ implies that $m(t) - \tilde{m}(t) \le [q(t) - \tilde{q}(t)]^+ \cdot \sqrt{\frac{\omega}{2\beta}} = 0$, which contradicts our assumption.

iii, a. Assume that $q(t) > \tilde{q}(t)$, $m(t) \le \tilde{m}(t)$, and $\hat{r}_1(t) = r_1$. However, $\hat{r}_1(t) = r_1$ implies that $m(t) - \tilde{m}(t) > [q(t) - \tilde{q}(t)] \cdot \sqrt{\frac{\omega}{2\beta}} \ge 0$, which contradicts our assumption.

iii, b. Assume that $q(t) > \tilde{q}(t)$, $m(t) \le \tilde{m}(t)$, and $\hat{r}_1(t) = R_1$. Note that $\hat{r}_2(t) = r_2$ because $m(t) \le \tilde{m}(t)$. The derivative of $d_{R_1}(t)$ is, therefore, at most

$$\begin{aligned}\beta \cdot \frac{\mathrm{d}d_{R_1}(t)}{\mathrm{d}t} &\le d_{R_1}(t)^{-1} \cdot \begin{pmatrix} R_1\omega \cdot (q(t) - \tilde{q}(t))(\lambda(t) - m(t)) \\ +R_1\omega \cdot (\tilde{q}(t) - q(t))(\lambda(t) - \tilde{m}(t)) \\ +(R_1\omega \cdot q(t) - r_2\theta \cdot m(t))(m(t) - \tilde{m}(t)) \\ +\beta \cdot \frac{\mathrm{d}\tilde{m}}{\mathrm{d}t} \cdot (\tilde{m}(t) - m(t)) \end{pmatrix} \\ &\le d_{R_1}(t)^{-1} \cdot (\tilde{m}(t) - m(t))\left(r_2\theta \cdot m(t) + \beta \cdot \frac{\mathrm{d}\tilde{m}}{\mathrm{d}t}\right) \\ &\le r_2\theta \cdot m(t) + \beta \cdot \left[\frac{\mathrm{d}\tilde{m}}{\mathrm{d}t}\right]^+.\end{aligned} \tag{A.82}$$

The derivative of the potential function $\Phi(t)$ is then

$$\begin{aligned}\frac{\mathrm{d}\Phi(t)}{\mathrm{d}t} &\le c_2 r_2\theta \cdot m(t) + c_2\beta \cdot \left[\frac{\mathrm{d}\tilde{m}}{\mathrm{d}t}\right]^+ + c_3\beta \cdot \frac{\mathrm{d}\tilde{m}}{\mathrm{d}t} \\ &\quad - \left(c_3 - \frac{1}{R_2}\right)\beta \cdot \frac{\mathrm{d}m}{\mathrm{d}t} + c_4\theta \cdot (\lambda(t) - m(t) - \lambda(t) + \tilde{m}(t)) \\ &\le c_2 r_2\theta \cdot m(t) + (c_2 + c_3)\beta \cdot \left[\frac{\mathrm{d}\tilde{m}}{\mathrm{d}t}\right]^+ \\ &\quad - \left(1 + \frac{1}{R_1}\right)\beta \cdot \frac{\mathrm{d}m}{\mathrm{d}t} + c_4\theta \cdot (\tilde{m}(t) - m(t)).\end{aligned} \tag{A.83}$$

The derivative of the cumulative cost $\textsc{Cost}(m,t)$ is

$$\frac{\partial \textsc{Cost}(m,t)}{\partial t} = \frac{\beta}{R_1} \cdot \frac{\mathrm{d}m}{\mathrm{d}t} + \beta \cdot \left[\frac{\mathrm{d}m}{\mathrm{d}t}\right]^+ + \left(1 + \frac{r_2}{R_1}\right)\theta \cdot m(t). \tag{A.84}$$

We sum Equations (A.83) and (A.84) and cancel terms to obtain

$$\frac{\mathrm{d}\Phi(t)}{\mathrm{d}t}+\frac{\partial \textsc{Cost}(m,t)}{\partial t}\le (c_2+c_3)\beta\cdot\left[\frac{\mathrm{d}\tilde{m}}{\mathrm{d}t}\right]^+ + c_4\theta\cdot\tilde{m}(t)$$
$$\le \mathrm{OCR}\cdot\frac{\partial \textsc{Cost}(\tilde{m},t)}{\partial t}. \tag{A.85}$$

iv, a. Assume that $q(t)\le\tilde{q}(t)$, $m(t)\le\tilde{m}(t)$, and $\hat{r}_1(t)=r_1$. However, $\hat{r}_1(t)=r_1$ implies that $m(t)-\tilde{m}(t)>[q(t)-\tilde{q}(t)]^+\cdot\sqrt{\frac{\omega}{2\beta}}=0$, which contradicts our assumption.

iv, b. Assume that $q(t)\le\tilde{q}(t)$, $m(t)\le\tilde{m}(t)$, and $\hat{r}_1(t)=R_1$. Note that $\hat{r}_2(t)=r_2$ because $m(t)\le\tilde{m}(t)$. The potential function $\Phi(t)$ simplifies to

$$\Phi(t)=(c_2+c_3)\beta\cdot(\tilde{m}(t)-m(t))+\frac{\beta\cdot m(t)}{R_2}. \tag{A.86}$$

The derivative of the potential function $\Phi(t)$ is then

$$\frac{\mathrm{d}\Phi(t)}{\mathrm{d}t}=(c_2+c_3)\beta\cdot\frac{\mathrm{d}\tilde{m}}{\mathrm{d}t}-\left(c_2+1+\frac{1}{R_1}\right)\beta\cdot\frac{\mathrm{d}m}{\mathrm{d}t}. \tag{A.87}$$

The derivative of the cumulative cost $\textsc{Cost}(m,t)$ is

$$\frac{\partial \textsc{Cost}(m,t)}{\partial t}=\frac{\beta}{R_1}\cdot\frac{\mathrm{d}m}{\mathrm{d}t}+\beta\cdot\left[\frac{\mathrm{d}m}{\mathrm{d}t}\right]^+ +\left(1+\frac{r_2}{R_1}\right)\theta\cdot m(t). \tag{A.88}$$

We sum Equations (A.87) and (A.88) and cancel terms to obtain

$$\frac{\mathrm{d}\Phi(t)}{\mathrm{d}t}+\frac{\partial \textsc{Cost}(m,t)}{\partial t}\le (c_2+c_3)\beta\cdot\left[\frac{\mathrm{d}\tilde{m}}{\mathrm{d}t}\right]^+ + c_4\theta\cdot\tilde{m}(t)$$
$$\le \mathrm{OCR}\cdot\frac{\partial \textsc{Cost}(\tilde{m},t)}{\partial t}. \tag{A.89}$$

□

A.11. Proof of Lemma 2

Proof of Lemma 2. We will construct a sequence of instances for which $\textsc{Opt}_{int}/\textsc{Opt}\to\infty$.

Let $m(t)$ be the number of servers of $\textsc{Opt}_{int}$. Fix $0<\varepsilon<1$, and let $\lambda(t)=\varepsilon$ for $t\in[0,T]$, where the finite-time horizon $T=1/\varepsilon$. Let $\beta=0$, $\omega=\infty$, and $\theta=1$. Then, because $\omega=\infty$, $m(t)\ge 1$ for $t\in[0,T]$. Therefore, $\textsc{Opt}_{int}\ge 1/\varepsilon$. However, one possible fractional solution turns on $m^*(t)=\varepsilon$ servers for $t\in[0,T]$, and therefore, the value of the optimal solution is at most $\textsc{Opt}\le 1$. Thus, $\textsc{Opt}_{int}/\textsc{Opt}=1/\varepsilon\to\infty$ as $\varepsilon\to 0$. □

References

[1] Albers S, Fujiwara H (2007) Energy-efficient algorithms for flow time minimization. *ACM Trans. Algorithms* 3(4):49.
[2] Albers S, Müller F, Schmelzer S (2014) Speed scaling on parallel processors. *Algorithmica* 68(2):404–425.
[3] Anderson C, Karlin AR (1996) Two adaptive hybrid cache coherency protocols. *Proc. Second Internat. Sympos. High-Performance Comput. Architecture* (IEEE, Piscataway, NJ), 303–313.
[4] Antoniadis A, Coester C, Elias M, Polak A, Simon B (2020) Online metric algorithms with untrusted predictions. *Internat. Conf. Machine Learn.* (PMLR), 345–355.
[5] Ata B, Shneorson S (2006) Dynamic control of an M/M/1 service system with adjustable arrival and service rates. *Management Sci.* 52(11):1778–1791.
[6] Augustine J, Irani S, Swamy C (2004) Optimal power-down strategies. *45th Annual IEEE Sympos. Foundations Comput. Sci.* (IEEE, Piscataway, NJ), 530–539.
[7] Azar Y, Bartal Y, Feuerstein E, Fiat A, Leonardi S, Rosén A (1999) On capital investment. *Algorithmica* 25(1):22–36.
[8] Bamas E, Maggiori A, Rohwedder L, Svensson O (2020) Learning augmented energy minimization via speed scaling. Preprint, submitted October 22, https://arxiv.org/abs/2010.11629.
[9] Bansal N, Chan HL, Pruhs K (2013) Speed scaling with an arbitrary power function. *ACM Trans. Algorithms* 9(2):1–14.
[10] Barbu V, Precupanu T (2012) *Convexity and Optimization in Banach Spaces* (Springer Science & Business Media, New York).
[11] Barroso LA, Hölzle U (2007) The case for energy-proportional computing. *Computer* 40(12):33–37.
[12] Bodik P (2010) Automating datacenter operations using machine learning. PhD thesis, University of California, Berkeley, CA.
[13] Boyar J, Favrholdt LM, Kudahl C, Larsen KS, Mikkelsen JW (2017) Online algorithms with advice: A survey. *ACM Comput. Surveys* 50(2):19.
[14] Cortez E, Bonde A, Muzio A, Russinovich M, Fontoura M, Bianchini R (2017) Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. *Proc. 26th Sympos. Operating Systems Principles*, 153–167.
[15] Damaschke P (2003) Nearly optimal strategies for special cases of on-line capital investment. *Theoret. Comput. Sci.* 302(1–3):35–44.
[16] Dayarathna M, Wen Y, Fan R (2015) Data center energy consumption modeling: A survey. *IEEE Comm. Surveys Tutorials* 18(1):732–794.
[17] Doytchinov B, Lehoczky J, Shreve S (2001) Real-time queues in heavy traffic with earliest-deadline-first queue discipline. *Ann. Appl. Probab.* 11(2):332–378.

[18] Facebook (2014) Making Facebook's software infrastructure more energy efficient with Autoscale. Accessed February 1, 2021, https://engineering.fb.com/production-engineering/making-facebook-s-software-infrastructure-more-energy-efficient-with-autoscale/.
[19] Galloway J, Smith K, Carver J (2012) An empirical study of power aware load balancing in local cloud architectures. *2012 Ninth Internat. Conf. Inform. Tech. New Generations* (IEEE, Piscataway, NJ), 232–236.
[20] Gandhi A, Doroudi S, Harchol-Balter M, Scheller-Wolf A (2013) Exact analysis of the M/M/k/setup class of Markov chains via recursive renewal reward. *Proc. ACM SIGMETRICS/Internat. Conf. Measurement Model. Comput. Systems* (ACM, New York), 153–166.
[21] Gandhi A, Dube P, Karve A, Kochut A, Zhang L (2014) Adaptive, model-driven autoscaling for cloud applications. *11th Internat. Conf. Autonomic Comput. (ICAC 14)*, 57–64.
[22] Gandhi A, Gupta V, Harchol-Balter M, Kozuch MA (2010) Optimality analysis of energy-performance trade-off for server farm management. *Performance Evaluation* 67(11):1155–1171.
[23] Gandhi A, Harchol-Balter M, Raghunathan R, Kozuch MA (2012) Autoscale: Dynamic, robust capacity management for multi-tier data centers. *ACM Trans. Comput. Systems* 30(4):14.
[24] Gao J (2014) Machine learning applications for data center optimization. Working paper, Google Research, New York.
[25] Goldman SA, Parwatikar J, Suri S (2000) Online scheduling with hard deadlines. *J. Algorithms* 34(2):370–389.
[26] Google (2016) Data centers get fit on efficiency. Accessed February 1, 2021, https://blog.google/outreach-initiatives/environment/data-centers-get-fit-on-efficiency/.
[27] Hsu CY, Indyk P, Katabi D, Vakilian A (2019) Learning-based frequency estimation algorithms. *Internat. Conf. Learn. Representations*.
[28] Irani S, Shukla S, Gupta R (2002) Competitive analysis of dynamic power management strategies for systems with multiple power saving states. *Proc. 2002 Design Automation Test Europe Conf. Exhibition* (IEEE, Piscataway, NJ), 117–123.
[29] Jones N (2018) How to stop data centres from gobbling up the world's electricity. *Nature* (September 12), Accessed February 1, 2021, https://www.nature.com/articles/d41586-018-06610-y.
[30] Karlin AR, Kenyon C, Randall D (2001) Dynamic TCP acknowledgment and other stories about e/(e − 1). *Proc. thirty-third annual ACM sympos. Theory comput.* (ACM, New York), 502–509.
[31] Karlin AR, Manasse MS, Rudolph L, Sleator DD (1988) Competitive snoopy caching. *Algorithmica* 3(1–4):79–119.
[32] Khanafer A, Kodialam M, Puttaswamy KP (2013) The constrained ski-rental problem and its application to online cloud cost optimization. *2013 Proc. IEEE INFOCOM* (IEEE, Piscataway, NJ), 1492–1500.
[33] Kumar R, Purohit M, Schild A, Svitkina Z, Vee E (2018) Semi-online bipartite matching. Preprint, submitted December 1, https://arxiv.org/abs/1812.00134v1.
[34] Lassettre E, Coleman DW, Diao Y, Froehlich S, Hellerstein JL, Hsiung L, Mummert T, et al. (2003) Dynamic surge protection: An approach to handling unexpected workload surges with resource actions that have lead times. *Internat. Workshop Distributed Systems Oper. Management* (Springer), 82–92.
[35] Lee R, Hajiesmaili MH, Li J (2019) Learning-assisted competitive algorithms for peak-aware energy scheduling. Preprint, submitted November 18, https://arxiv.org/abs/1911.07972.
[36] Lin M, Wierman A, Andrew LL, Thereska E (2012) Dynamic right-sizing for power-proportional data centers. *IEEE/ACM Trans. Networking* 21(5):1378–1391.
[37] Lu T, Chen M, Andrew LL (2012) Simple and effective dynamic provisioning for power-proportional data centers. *IEEE Trans. Parallel Distributed Systems* 24(6):1161–1171.
[38] Lykouris T, Vassilvtiskii S (2021) Competitive caching with machine learned advice. *J. ACM* 68(4):1–25.
[39] Maccio VJ, Down DG (2015) On optimal policies for energy-aware servers. *Performance Evaluation* 90:36–52.
[40] Mahdian M, Nazerzadeh H, Saberi A (2012) Online optimization with uncertain information. *ACM Trans. Algorithms* 8(1):2.
[41] Manmeet S, Maninder S, Sanmeet K (2019) TI-2016 DNS dataset. IEEE DataPort. Accessed January 4, 2021, https://ieee-dataport.org/documents/ti-2016-dns-dataset.
[42] Mazzucco M, Dyachuk D (2012) Optimizing cloud providers revenues via energy efficient server allocation. *Sustainable Comput. Informatics Systems* 2(1):1–12.
[43] Mitzenmacher M (2018) A model for learned bloom filters and optimizing by sandwiching. *32nd Conf. Neural Inform. Processing Systems 31 (NeurIPS 2018)*, 464–473.
[44] Mitzenmacher M (2019a) Scheduling with predictions and the price of misprediction. Preprint, submitted May 23, https://arxiv.org/abs/1902.00732.
[45] Mitzenmacher M (2019b) The supermarket model with known and predicted service times. Preprint, submitted May 23, https://arxiv.org/abs/1905.12155v1.
[46] Mukherjee D, Stolyar A (2019) Join idle queue with service elasticity: Large-scale asymptotics of a nonmonotone system. *Stochastic Systems* 9(4):338–358.
[47] Mukherjee D, Dhara S, Borst SC, van Leeuwaarden JSH (2017) Optimal service elasticity in large-scale distributed systems. *Proc. ACM Measurement Anal. Comput. Systems* 1(1):1–28.
[48] Netflix (2013) Scryer: Netflix's predictive auto scaling engine. Accessed February 1, 2021, https://netflixtechblog.com/scryer-netflixs-predictive-auto-scaling-engine-a3f8fc922270.
[49] Purohit M, Svitkina Z, Kumar R (2018) Improving online algorithms via ml predictions. *Adv. Neural Inform. Processing Systems 31 (NeurIPS 2018)*, 9661–9670.
[50] Qi J, Du J, Siniscalchi SM, Ma X, Lee CH (2020) On mean absolute error for deep neural network based vector-to-vector regression. *IEEE Signal Processing Lett.* 27:1485–1489.
[51] Rong H, Zhang H, Xiao S, Li C, Hu C (2016) Optimizing energy consumption for data centers. *Renewable Sustainable Energy Rev.* 58(C):674–691.
[52] Rzadca K, Findeisen P, Swiderski J, Zych P, Broniek P, Kusmierek J, Nowak P, et al. (2020) Autopilot: Workload autoscaling at Google. *Proc. Fifteenth Eur. Conf. Comput. Systems*, 1–16.
[53] Shehabi A, Smith S, Sartor D, Brown R, Herrlin M, Koomey J, Masanet E, Horner N, Azevedo I, Lintner W (2016) United States data center energy usage report. Technical report, Lawrence Berkeley National Laboratory, Berkeley, CA.

[54] Sverdlik Y (2020) How Zoom, Netflix, and Dropbox are staying online during the pandemic. Accessed February 1, 2021, https://www.datacenterknowledge.com/uptime/how-zoom-netflix-and-dropbox-are-staying-online-during-pandemic.
[55] Tirmazi M, Barker A, Deng N, Haque ME, Qin ZG, Hand S, Harchol-Balter M, Wilkes J (2020) Borg: The next generation. *Proc. Fifteenth Eur. Conf. Comput. Systems*, 1–14.
[56] Urgaonkar B, Shenoy P, Chandra A, Goyal P (2005) Dynamic provisioning of multi-tier internet applications. *Second Internat. Conf. Autonomic Comput. (ICAC'05)* (IEEE, Piscataway, NJ), 217–228.
[57] Wierman A, Andrew LL, Tang A (2009) Power-aware speed scaling in processor sharing systems. *2009 Proc. IEEE INFOCOM* (IEEE, Piscataway, NJ), 2007–2015.