

Dynamic Capacity Provisioning of Heterogeneous Servers

Minghong Lin*, Lachlan L. H. Andrew†, Adam Wierman*

*California Institute of Technology, Email: {mhlin,adamw}@caltech.edu

†Swinburne University of Technology, Email: landrew@swin.edu.au

Abstract—An important technique for saving energy in data centers is consolidating work onto a small number of well utilized servers via dynamic capacity provisioning. A common approach for dynamic capacity provisioning is “receding horizon control” (RHC), which computes the provisioning for the current time by optimizing over a given window of predictions about the future arrivals. In this work, we provide new results characterizing the performance of RHC. We prove that RHC performs well when servers are homogeneous; specifically, it has performance that quickly tends toward optimality as the prediction window increases. However, we also prove that RHC can perform badly when servers are heterogeneous, regardless of the length of the prediction window. This is problematic since in practice there is heterogeneity both within and between data centers. To address this issue, we introduce two variants of RHC that are guaranteed to perform well in heterogeneous settings. Specifically, under both homogeneous and heterogeneous settings, their competitive ratio matches that of RHC in the homogeneous setting.

I. INTRODUCTION

As the Internet continues to grow, it is increasingly hampered by excess energy consumption. This is particularly acute in data centers, which can now use as much power as a small city [1]. However, much of this power is wasted on times when load is low. There are two main techniques to reduce energy consumption during periods of low load: speed scaling [1]–[3] and low-power idle (sleep) modes [4]–[7]. We focus on sleep. It is often argued that it is better to use “energy proportional” servers than to put servers to sleep, but much progress towards making servers energy proportional has come from making them autonomously put subsystems to sleep.

Whether autonomous or centralized, entering and leaving sleep mode incurs a penalty (“switching cost”), in terms of latency, energy consumption, or wear-and-tear. This means that decisions to sleep cannot be made independently at different time instants. The problem is challenging due to the lack of knowledge about future workloads, which means that a server that is put to sleep may soon need to be woken again. There is a significant and growing literature on this topic [5],[6],[8],[9].

Although the distant future cannot be known, it is often possible to estimate loads a little in the future [10], [11]. This can be used by Receding Horizon Control (RHC) algorithms, also known as Model Predictive Control. RHC is commonly proposed to control data centers [8], [9] and, more broadly, it has a long history in the control theory literature [12]–[14]. In RHC, an estimate of the near future is used to design a tentative control trajectory; the first step of this trajectory is then implemented and, in the next time step, a new prediction and new tentative trajectory are calculated.

In this paper, we evaluate bounds on the performance of RHC for controlling the number of servers to provision in a data center. The performance is in terms of the competitive ratio: the worst case ratio of the cost of using RHC to the cost of using optimal provisioning.

We first consider the case that all servers are equally able to serve each job. In this case, RHC is $1 + O(1/w)$ -competitive when load can be accurately estimated w into the future. This can be much tighter than the competitive ratio of 3 obtained for “blind” schemes that know nothing of future loads [6].

However, in many settings, the cost of serving a job differs between servers. Even if their hardware is identical, servers within a data center may store different data on their local hard drives. Similarly, geographic load balancing [15], [4] causes work to be sent to distant servers, which incurs unequal network delay costs. Within a data center, servers may be different generations of hardware, or deliberately chosen to be heterogeneous, possibly to increase the flexibility of power management [16]–[18].

In the heterogeneous setting, we find that there are workloads for which RHC performs poorly. In particular, it is $1 + \Omega(\beta/f_0)$ -competitive, where β measures the switching cost and f_0 is the cost of running an idle server. This can be large and, surprisingly, does not depend on w . That is, in the worst case, RHC does not improve as the prediction window grows.

Motivated by this weakness, we present variants of RHC that maintain $1 + O(1/w)$ worst-case guarantees regardless of the number of types of servers and workloads. The new algorithms we propose work by taking the average or mixture of $w + 1$ Fixed Horizon Control (FHC) algorithms. See Section III for a detailed description. In addition to providing analytic results for these algorithms, we evaluate them under workloads measured on a real data center, and show that the improvement in worst-case performance comes at virtually no cost in average-case performance.

Note that one distinctive feature of this paper is the generality of the model we consider. In addition to allowing heterogeneity among both the jobs and the servers (systems studied analytically typically have homogeneous servers [5]–[7] or disjoint collections of homogeneous servers [19]); we also allow costs to be general, i.e., we do not require convexity (typically analytic guarantees require convex cost functions, e.g. [4]–[6],[15]).

The remainder of the paper is organized as follows. After describing the analytical model in Section II, we present our algorithms in Section III, where we also discuss their performance. The results are proved in Section IV.

II. MODEL

Our focus is on understanding how to dynamically provision the (active) service capacity of a large, possibly heterogeneous pool of servers so as to minimize the “cost” of the system, which may include both energy and quality of service. The model we discuss generalizes [6],[4],[15] and can, for example, model servers in geographically diverse data centers serving requests from different regions, or servers storing different files and serving jobs requiring different data.

A. Workload model

We consider $S \geq 1$ types of servers, each of which has a different cost for serving different types of jobs, and $J \geq 1$ types of jobs. We take a discrete-time model where the timeslot length matches the timescale at which servers can enter or leave power saving states. There is a (possibly long) time-interval of interest $t \in \{1, \dots, T\}$. The mean request rate for timeslot t is denoted by $\lambda_t = (\lambda_{t,j})_{j \in J}$, where $\lambda_{t,j}$ is the mean request rate (arrival rate) for type j jobs at time t . We set $\lambda_t = 0$ for $t < 1$ and $t > T$, and assume that jobs are short so that work does not carry over between slots and provisioning can be based on the average arrival rate during a timeslot. In the data center setting, T could be a year, a timeslot could be 10 minutes, and a job length could be on a second or less.

B. Cost model

Our goal is to provide insight into two important decisions:

- (i) Determining the number of active servers, $x_t = (x_{t,s})_{s \in S}$, where $x_{t,s}$ is the number of active servers of type s during timeslot t . Define $x = (x_t)_{t=1}^T$.
- (ii) Dispatching arrivals to servers, i.e., determining $(\lambda_{t,j,s})_{s \in S}$ such that $\sum_{s \in S} \lambda_{t,j,s} = \lambda_{t,j}$ for all $j \in J$.

The system wants to choose the number of active servers and the dispatching in order to minimize the “cost” incurred.

We decompose the cost incurred by the system into two components: (i) the *operating cost* incurred by using active servers to serve requests by certain dispatching rule in each timeslot. (ii) the *switching cost* incurred by changing provisioning between timeslots. Note that both components may include costs of energy, delay, and even wear-and-tear.

To model the operating costs, we use a (possibly time varying) function $f_t(x_t, \lambda_t)$ for each timeslot, which represents the cost of using $x_t = (x_{t,1}, \dots, x_{t,S})$ servers to serve arriving requests $\lambda_t = (\lambda_{t,1}, \dots, \lambda_{t,J})$ under the optimal dispatching of λ_t over x_t . Note that the optimal dispatching can be computed easily in many cases, such as when each f_t is convex. Thus, we do not explicitly consider the dispatching decision in the following and simply assume it is performed optimally.

One important property of f for our results is $f_{0,s}$, the minimum cost per timeslot for an active server of type s . Thus $f_t(x_t, \lambda_t) \geq f_0 \cdot x_t$ where $f_0 = (f_{0,s})_{s \in S}$.

The switching costs are modeled by a function $d(x_{t-1}, x_t)$, which represents the cost of changing the number of active servers of each type from vector x_{t-1} to vector x_t . Except for Lemmas 1 and 2, our results use the following switching cost. Let β^+ and β^- be vectors such that β_s^+ is the cost of

turning a server of type s on, and β_s^- is the cost of turning it off. Then, it is natural to use

$$d(x, y) = \beta^+ \cdot (y - x)^+ + \beta^- \cdot (x - y)^+, \quad (1)$$

where $(x)^+ = \max(0, x)$ elementwise.

C. Cost optimization problem

Given the workload and cost models above, the system goal is to choose the active number of servers x_t so as to minimize the total cost during $[1, T]$:

$$\begin{aligned} \min_{x_1, \dots, x_T} \quad & \sum_{t=1}^T f_t(x_t, \lambda_t) + \sum_{t=1}^{T+1} d(x_{t-1}, x_t) \\ \text{subject to} \quad & 0 \leq x_t \in \mathbb{R}^J, \quad x_0 = x_{T+1} = 0. \end{aligned} \quad (2)$$

Recall that x_t and λ_t are vectors. Note that this optimization makes two main simplifications. First, it does not impose integer constraints on x_t . This is acceptable since the number of servers is assumed to be large and so rounding does not create significant inefficiency. Second, the number of each type of server is not explicitly bounded above. A (possibly time-varying) upper bound $x_t \leq M_t$ can be imposed by defining $f_t(x, \cdot) = \infty$ for $x > M_t$. A constraint on the load per server can be imposed similarly.

In the optimization above, since $x_0 = 0$ and $x_{T+1} = 0$, the number of times a server is turned on in $[1, T]$ is equal to the number of times it is turned off in $[2, T+1]$. Thus, if the switching costs are modeled using (1) then the solution of (2) depends on β^+ and β^- only through their sum. Accordingly, for simplicity, we focus on

$$d(x, y) = \beta \cdot (y - x)^+, \quad (3)$$

where $\beta = \beta^+ + \beta^-$. In this case, the last switching cost is 0, and the optimization becomes

$$\begin{aligned} \min_{x_1, \dots, x_T} \quad & \sum_{t=1}^T f_t(x_t, \lambda_t) + \sum_{t=1}^T d(x_{t-1}, x_t) \\ \text{subject to} \quad & 0 \leq x_t \in \mathbb{R}^J, \quad x_0 = 0. \end{aligned} \quad (4)$$

It is this formulation of the cost optimization with d as in (3) that we focus on in the remainder of the paper, unless otherwise stated.¹

Given this optimization problem, in many cases the solution can be found easily *offline*, i.e., given λ_t for all t . However, our goal is to find *online* algorithms for this optimization, i.e., algorithms that determine x_t using only information up to time $t + w$ where $w \geq 0$ is called the “prediction window”.

In order to evaluate the performance of online algorithms we use the standard notion of *competitive ratio*. The competitive ratio of an algorithm \mathcal{A} is defined as the maximum, taken over all possible inputs, of $\text{cost}(\mathcal{A})/\text{cost}(\text{OPT})$, where $\text{cost}(\mathcal{A})$ is the objective function of (4) under algorithm \mathcal{A} and OPT is the optimal offline algorithm. In our context, the “input”

¹The fact that the choice of β^+ and β^- does not affect the solution of (2) allows algorithms to choose an arbitrary β^+ and β^- subject to $\beta^+ + \beta^- = \beta$. In the algorithms we study, $\beta^- = 0$ is the optimal choice.

clearly includes the load λ and duration T , and we also take it to include S , J and β .

A final remark about optimizations (2) and (4) is that, when $d(x, y)$ is a metric, they are instances of the class of problems known as ‘‘Metrical Task Systems (MTSs)’’.² MTSs have received considerable study in the algorithms literature, and it is known that if no further structure is placed on them, then the best deterministic algorithm for a MTS has competitive ratio proportional to the number of system states [20]. However, when additional structure exists the competitive ratio can be much smaller [21], [22]. The work most related to the current paper is [6], which gave a 3-competitive algorithm for a special case of (4) where servers are homogeneous, f is convex, and the prediction window $w = 0$.

In this paper, we study both homogeneous and heterogeneous systems and allow f to be general. In this setting, the prediction window becomes a key determinant of the performance of the online algorithms we consider. For some MTSs, prediction windows of fixed length provide no benefit in terms of the competitive ratio [23]. However, in our setting, we show that significant benefits are possible.

D. Motivating examples

We now highlight the generality of the model by providing a few examples of problems that it captures.

Geographical load balancing: The first example we consider is that of load balancing among geographically diverse data centers in order to minimize the cost, which depends on time-varying electricity prices at each of the locations. This setting has been the focus of a considerable amount of algorithmic work recently, e.g., [4],[24],[25].

To capture this scenario, the job types represent jobs originating at geographically different sources and the server types represent servers at geographically different data centers. The operating costs, f , then include the network delays between sources and servers, queueing/processing delays within each server, and energy costs for operating servers in each location.

For example, each server may be modeled as an M/G/1/PS (processing sharing) queue. Let λ_{js} be the arrival rate jobs from location j to data center s , and $f_{0,s}$ be the energy cost of a server of type s . Let the network delay from location j to data center s be D_{js} . If the service rate of each server is normalized to $\mu = 1$, then $f(x, \lambda)$ is

$$\min_{\lambda_{js}} \sum_{s=1}^S \left(x_s f_{0,s} + \frac{(\sum_j \lambda_{sj})^2}{1 - \sum_j \lambda_{sj}/x_s} + \sum_j \lambda_{sj} D_{js} \right) \quad (5)$$

s.t. $\sum_s \lambda_{sj} = \lambda_j.$

The switching costs, d , in this scenario include a combination of the power, time, and wear-and-tear involved in turning a server on/off. Thus, $d(x, y) = \beta(y - x)^+$ is reasonable.

²Note that we do not restrict $d(\cdot)$ to being a metric. In particular, we focus on d , for which $d(x, y) = 0$ does not imply $x = y$. However, all problems we consider are equivalent to ones in which a metric $d(\cdot)$ does exist, namely setting $\beta^+ = \beta^- > 0$.

A heterogeneous data center: Much of the research on dynamic capacity provisioning in data centers, e.g., [7], [5], has assumed homogeneity, with servers equally able to serve all jobs. The model we consider allows a much more complex situation where each job type may have different service rates at different types of servers. This could result from heterogeneous data placement or hardware, among other causes, e.g., heterogeneity may be deliberately introduced to facilitate energy management [16]–[18].

In this setting, the job types require different mixes of data or have different bottleneck resources. The server types capture heterogeneous server hardware (such as different types of cores on a heterogeneous multicore CPU [26] or amounts of memory) or data placement. The operating costs, f , include queueing/processing delays and the communication time to access data, which may be stored remotely.

For example, as in the geographical load balancing case, each server may be modeled as an M/G/1/PS queue, and each job of type j impose load c_{js} on a server of type s . Let λ_{js} be the arrival rate of type j jobs to all the servers of type s , and other notation be as above. Then (5) is replaced by

$$f(x, \lambda) = \min_{\lambda_{js}} \sum_{s=1}^S \left(x_s f_{0,s} + \frac{(\sum_j \lambda_{js} c_{js})^2}{1 - (\sum_j \lambda_{js} c_{js})/x_s} \right). \quad (6)$$

The switching cost contains the same components as above.

III. ALGORITHMS AND RESULTS

Our focus in this paper is on evaluating analytically the performance of Receding Horizon Control (RHC) as an algorithm for dynamic capacity provisioning. Our results uncover some potential drawbacks of RHC in heterogeneous settings, and so we also propose variations of RHC that address these drawbacks. The main results are presented and discussed here. The proofs are deferred to Section IV.

A. Receding Horizon Control (RHC)

As previously mentioned, RHC is commonly proposed in for dynamic capacity provisioning [8], [9] and has a long history in the control theory literature [12]–[14].

Informally, RHC works by, at time τ , solving the cost optimization over the window $(\tau, \tau + w)$ given the starting state $x_{\tau-1}$. Formally, define $X^\tau(x_{\tau-1}; \lambda)$ as the vector in $(\mathbb{R}^J)^{w+1}$ indexed by $t \in \{\tau, \dots, \tau + w\}$, which is the solution to

$$\min_{x_\tau, \dots, x_{\tau+w}} \sum_{t=\tau}^{\tau+w} f_t(x_t, \lambda_t) + \sum_{t=\tau}^{\tau+w} d(x_{t-1}, x_t) \quad (7)$$

subject to $x_t \geq 0$.

Then, RHC works as follows.

Algorithm 1 (Receding Horizon Control: RHC). *For all $t \leq 0$, set the number of active servers to $x_{RHC,t} = 0$. At each timeslot $\tau \geq 1$, set the number of active servers to*

$$x_{RHC,\tau} = X_\tau^\tau(x_{RHC,\tau-1}; \lambda) \quad (8)$$

and optimally dispatch λ_τ across $x_{RHC,\tau}$.

To study the performance of RHC we first consider the case of *homogeneous* servers ($S = 1$). Then, x_t are scalars. This case still has jobs with different service requirements, and so λ_t may still be a vector. Note that (7) need not have a unique solution. In the homogeneous case, we define RHC to select the solution with the greatest first entry.

In the homogeneous setting, RHC performs well: it has a small constant competitive ratio that depends on the minimal cost of a server and switching cost, and decays to 1 quickly as the prediction window grows.

Theorem 1. *If there is a single type of server ($S = 1$), then RHC is $(1 + \frac{\beta}{(w+1)f_0})$ -competitive.*

Given Theorem 1, it is natural to wonder if the competitive ratio is tight. The following result highlights that there exist settings where the performance of RHC is quite close to the bound in Theorem 1.

Theorem 2. *There exists a (convex) operating cost function f such that RHC is not better than $(\frac{1}{w+2} + \frac{\beta}{(w+2)f_0})$ -competitive, even when restricted to $S = 1$.*

Note that Theorem 2 holds even if we restrict ourselves to the class of convex objectives. Indeed, all of the examples of bad performance in this paper can be achieved with convex cost functions. Additionally, it is interesting to note that [27] shows that a prediction window of w can improve the performance of a metrical task system by a factor of at most $2w$. If $\beta/f_0 \gg 1$ then RHC is approximately within a factor of 2 of this limit in the homogeneous case.

The two theorems above highlight that, with enough lookahead, RHC is guaranteed to perform quite well in the homogeneous case. Unfortunately, the story is different when servers are *heterogeneous*.

Theorem 3. *When there are multiple types of servers ($S \geq 2$), for all $w \geq 0$ RHC is $\geq (1 + \max_s(\beta_s/f_{0,s}))$ -competitive.*

In particular, for all $w > 0$ the competitive ratio in the heterogeneous case is at least as large as the competitive ratio in the homogeneous case with $w = 0$. Most surprisingly (and problematically) this highlights that RHC may not see any improvement in the competitive ratio as w is allowed to grow. Further, this worst case uses convex f , so the hardness is truly coming from the heterogeneity and not from other factors.

The proof, given in Section IV-D involves constructing a workload such that servers of different types turn on and off in a cyclic fashion under RHC, whereas the optimal solution is to use a single class of servers throughout. Note that the larger the prediction window w is, the larger the number of types of servers must be in order to achieve this worst case.

The reason that RHC performs poorly in the heterogeneous case is that it neglects history, i.e., how long it has been in the current state. At first glance, this seems reasonable since the future cost depends on the history only through the state x_t . However, the cost of the optimal algorithm depends on a more detailed history and, by considering the history, an algorithm is able to reduce its competitive ratio. This is common in on-

line decision problems, such as the rent-or-buy problem [28] and k -server problem [29]. The algorithms in the next section address this issue.

B. Fixed Horizon Control

In this section, we present variants of RHC that collectively we term Fixed Horizon Control (FHC) which addresses the limitation of RHC in the heterogeneous setting.

Intuitively, instead of considering only the first step of each tentative control trajectory, at time t FHC takes the average of all the policies for time t calculated in preceding time steps.

More formally, first consider a family of algorithms parameterized by $k \in [1, w + 1]$ that recompute their provisioning periodically. For all $k = 1, \dots, w + 1$, let $\Omega_k = \{i : i \equiv k \pmod{w+1}\} \cap [-w, \infty)$; this is the set of integers congruent to k modulo $w + 1$, such that the lookahead window at each $\tau \in \Omega_k$ contains at least one $t \geq 1$.

Algorithm 2 (Fixed Horizon Control, version k : FHC^(k)). *For all $t \leq 0$, set the number of active servers to $x_{FHC,t}^{(k)} = 0$. At each timeslot $\tau \in \Omega_k$, for all $t \in \{\tau, \dots, \tau + w\}$, set*

$$x_{FHC,t}^{(k)} = X_t^\tau \left(x_{FHC,\tau-1}^{(k)}; \lambda \right) \quad (9)$$

using (7), and dispatch λ_t over $x_{FHC,t}^{(k)}$ optimally.

For notational convenience, we often set $x^{(k)} \equiv x_{FHC}^{(k)}$. Note that for $k > 1$ the algorithm must start from $\tau = k - (w + 1)$ rather than $\tau = k$ in order to calculate $x_{FHC,t}^{(k)}$ for $t < k$. However, $X_t^\tau(\cdot; \lambda) = 0$ for all $t < 0$ since $\lambda_t = 0$ for $t < 0$, and so the assignment in (9) is consistent with the initialization $x_{FHC,t}^{(k)} = 0$ for $t < 0$.

The above algorithm can have very poor performance. However, it gives rise to the following two useful algorithms, SFHC and AFHC, which provide ways to combine the $w + 1$ FHC algorithms to ensure good performance.

The first algorithm, Splitting FHC (SFHC), works by splitting the servers into $w + 1$ groups (each using a different FHC algorithm) and splitting the arrivals among them.

Algorithm 3 (Splitting FHC: SFHC). *Divide the servers into $w + 1$ groups and divide the workload equally among them. At timeslot $\tau \in \Omega_k$, group k uses FHC^(k) with load $\lambda/(1+w)$ to determine its provisioning $\hat{x}_\tau^{(k)}, \dots, \hat{x}_{\tau+w}^{(k)}$ and the corresponding optimal dispatching of $\lambda/(1+w)$ within group k during $[\tau, \tau + w]$.*

Note that at time $t \in \Omega_k$, groups $k' \neq k$ do not change their provisioning and dispatching. This causes the total provisioning at time t to depend on past loads as well as the current and future load. This is what allows SFHC to achieve a tighter competitive ratio than RHC.

The second algorithm we consider averages the decisions of the $w + 1$ FHC algorithms.

Algorithm 4 (Averaging FHC: AFHC). *At timeslot $\tau \in \Omega_k$, use FHC^(k) to determine the provisioning $x_{k,\tau}^{(k)}, \dots, x_{k,\tau+w}^{(k)}$ and then set $x_{AFHC,t} = \sum_{k=1}^{w+1} x_{k,t}^{(k)}/(w+1)$. At time t , λ_t is dispatched optimally over $x_{AFHC,t}$.*

These two algorithms are quite similar, and under a natural condition (see (10) below), both SFHC and AFHC give the same provisioning x_t . But, it should be noted that AFHC uses the globally optimal dispatching and SFHC does not; therefore AFHC outperforms SFHC. However, the splitting performed by SFHC facilitates distributed implementation.

Intuitively, the SFHC and AFHC seem worse than RHC because RHC uses the latest information to make the current decision and SFHC/AFHC make decisions in advance, thus ignoring some possibly valuable information.³ This intuition is partially true, as shown in the following theorem, which states that RHC is not worse than SFHC/AFHC for any workload in a homogeneous system ($S = 1$).

Theorem 4. *If there is a single type of server ($S = 1$), then $\text{cost}(RHC) \leq \text{cost}(AFHC) \leq \text{cost}(SFHC)$*

Though RHC is always better than AFHC and SFHC in the homogeneous case, RHC can be worse than SFHC and AFHC in heterogeneous systems, even when there are only two types of servers.

Theorem 5. *Consider $S = 2$ types of servers and $J = 1$ type of workload. There exists a (convex) running cost f and a workload λ such that*

$$\text{cost}(RHC) > \text{cost}(SFHC) \geq \text{cost}(AFHC).$$

Moreover, the competitive ratio of RHC can be much worse than that of SFHC and AFHC in larger heterogeneous systems, which highlights the importance of history to online algorithms. The following two theorems highlight, under slightly different conditions, that SFHC and AFHC guarantee good performance in the heterogeneous setting.

Theorem 6. *If*

$$f_t(\alpha x_t, \alpha \lambda_t) = \alpha f_t(x_t, \lambda_t), \quad (10)$$

for all $\alpha > 0$, then

$$\frac{\text{cost}(AFHC)}{\text{cost}(OPT)} \leq \frac{\text{cost}(SFHC)}{\text{cost}(OPT)} \leq 1 + \max_s \frac{\beta_s}{(w+1)f_{0,s}}.$$

Theorem 7. *If*

$$f_t(x_t, \lambda_t) \text{ is convex in } x_t, \quad (11)$$

then

$$\frac{\text{cost}(AFHC)}{\text{cost}(OPT)} \leq 1 + \max_s \frac{\beta_s}{(w+1)f_{0,s}}.$$

The contrast between Theorem 3 and Theorems 6 and 7 highlights the improvement AFHC and SFHC provide over RHC. In fact, AFHC and SFHC have the same competitive ratio in the general (possibly heterogeneous) setting that RHC has in the homogeneous setting.

³SFHC and AFHC are also more sensitive to errors in predicted workload than RHC is. The former “lock in” estimates made w steps in advance, whereas in the latter the provisioning at each time t reflects the most up-to-date estimates. The sensitivity can be reduced by re-calculating $x_t^{(k)}$ for all k based on the load estimates available at time t .

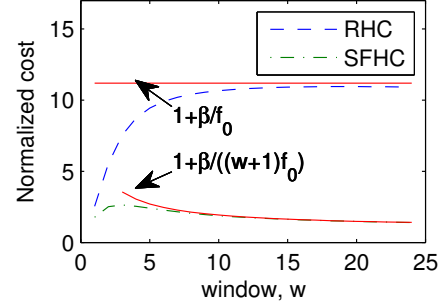


Fig. 1. Total cost, normalized by the cost of OPT, versus prediction window under RHC and SFHC for the workload used in the proof of Theorem 3.

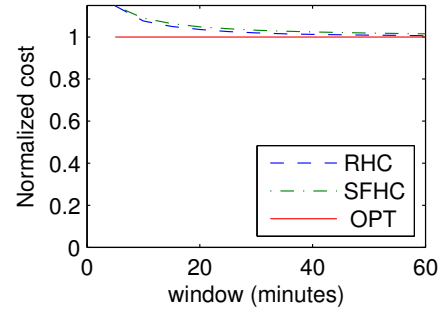


Fig. 2. Total cost, normalized by the cost of OPT, versus prediction window under RHC and AFHC for a workload trace from HP labs. Case $w = 0$ uses 5 minute lookahead to know the load in the “current” slot.

C. Numerical results

We highlight the improvement of AFHC and SFHC as compared to RHC in Figure 1, which illustrates the performance of SFHC, and RHC on a “bad” instance for RHC. This instance is described in the proof of Theorem 3. Notice that while SFHC (and AFHC) benefit from increasing window size, RHC fails to improve as the window size grows.

One might worry that, though AFHC and SFHC have significantly better performance in the worst-case than RHC, they might provide significantly worse performance in typical cases due to the fact that the latter always uses the latest information to make decisions while AFHC and SFHC do not. Figure 2 shows that this is not the case. In fact, in a typical setting, all three algorithms have quite similar cost. So, the “robustness” benefits of AFHC and SFHC do not come at significant expense to the average case. This setting uses the cost (6) with two types of server and two types of load; Type 1 servers are 30 times better at serving type 1 load than type 2 load, and conversely for type 2 servers. Loads are taken from an HP research labs data center, as described in [4], with 5 minute granularity. To introduce heterogeneity between different types of loads, the load of type 1 is the concatenation of load from a weekday followed by the load from a Sunday, while load of type 2 is that from a Sunday followed by a weekday. The switching cost is the energy cost of remaining idle for an hour.

IV. PROOFS

We now present the proofs of the theorems in Section III. The exposition is simplified by studying Fixed Horizon Control (FHC) before Receding Horizon Control (RHC).

A. Notation

We first introduce some additional notation used in the proofs. For brevity, for any vector y we write $y_{i..j} = (y_i, \dots, y_j)$ for any $i \leq j$.

Let x^* denote the offline optimal solution to optimization (4), and OPT be the algorithm that uses x^* with optimal dispatching at each stage. Further, let X be the result of RHC, and recall that $x^{(k)}$ is the result of FHC^(k), and $\hat{x}^{(k)}$ ($k = 1, \dots, w+1$) is the result for group k of SFHC.

Let the cost during time $[t_1, t_2]$ with optimal dispatching be

$$g_{t_1, t_2}(x; \xi, \lambda) = \sum_{t=t_1}^{t_2} f_t(x_t, \lambda_t) + d(\xi, x_{t_1}) + \sum_{t=t_1+1}^{t_2} d(x_{t-1}, x_t). \quad (12)$$

If ξ is omitted, then by convention $\xi = x_{t_1-1}$. Note that $g_{t_1, t_2}(x; \lambda)$ depends on x_i only for $t_1 - 1 \leq i \leq t_2$.

Using g we can define the cost of each algorithm as follows. For any algorithm $A \in \{RHC, FHC, AFHC, OPT\}$, which all use optimal dispatching at each stage, the total cost is

$$\text{cost}(A) = g_{1, T}(x_A; \lambda).$$

Since SFHC uses suboptimal dispatching, its total cost is slightly different, namely

$$\text{cost}(SFHC) = \sum_{k=1}^{w+1} g_{1, T}\left(\hat{x}^{(k)}; \frac{\lambda}{1+w}\right). \quad (13)$$

B. Proofs of Theorems 6 and 7

We first present lemmas needed to prove Theorem 6.

Lemma 1. For any $d(\cdot)$ that satisfies triangle inequality,

$$\text{cost}(FHC^{(k)}) \leq \text{cost}(OPT) + \sum_{\tau \in \Omega_k} d(x_{\tau-1}^{(k)}, x_{\tau-1}^*).$$

Proof: For every $k = 1, \dots, w+1$ and every $\tau \in \Omega_k$,

$$\begin{aligned} & g_{\tau, \tau+w}(x^{(k)}; \lambda) \\ &= \sum_{t=\tau}^{\tau+w} f_t(x_t^{(k)}, \lambda_t) + \sum_{t=\tau}^{\tau+w} d(x_{t-1}^{(k)}, x_t^{(k)}) \\ &\leq \sum_{t=\tau}^{\tau+w} f_t(x_t^*, \lambda_t) + \sum_{t=\tau+1}^{\tau+w} d(x_{t-1}^*, x_t^*) + d(x_{\tau-1}^{(k)}, x_{\tau}^*) \\ &\leq \sum_{t=\tau}^{\tau+w} f_t(x_t^*, \lambda_t) + \sum_{t=\tau+1}^{\tau+w} d(x_{t-1}^*, x_t^*) \\ &\quad + d(x_{\tau-1}^{(k)}, x_{\tau-1}^*) + d(x_{\tau-1}^*, x_{\tau}^*) \\ &= g_{\tau, \tau+w}(x^*; \lambda) + d(x_{\tau-1}^{(k)}, x_{\tau-1}^*). \end{aligned} \quad (14)$$

Summing the above over $\tau \in \Omega_k$, establishes the lemma. ■

Lemma 2. For every $f(\cdot)$ and $d(\cdot)$ such that for all x, y, λ and $\alpha > 0$,

$$f_t(\alpha x, \alpha \lambda) = \alpha f_t(x, \lambda),$$

$$d(\alpha x, \alpha y) = \alpha d(x, y)$$

and $d(\cdot)$ satisfies the triangle inequality, we have

$$\text{cost}(SFHC) \leq \text{cost}(OPT) + \sum_{k=1}^{w+1} \sum_{\tau \in \Omega_k} d\left(\hat{x}_{\tau-1}^{(k)}, \frac{x_{\tau-1}^*}{w+1}\right).$$

Proof: The assumptions on $f(\cdot)$ and $d(\cdot)$ imply that $g_{1, T}(\alpha x, \alpha \lambda) = \alpha g_{1, T}(x, \lambda)$. Since $\hat{x}^{(k)} = x^{(k)}/(w+1)$, substituting Lemma 1 into the definition (13) gives

$$\begin{aligned} \text{cost}(SFHC) &= \sum_{k=1}^{w+1} g_{1, T}(\hat{x}^{(k)}; \lambda/(w+1)) \\ &\leq \sum_{k=1}^{w+1} \left(g_{1, T}\left(\frac{x^*}{w+1}; \frac{\lambda}{w+1}\right) \right. \\ &\quad \left. + \sum_{\tau \in \Omega_k} d\left(\hat{x}_{\tau-1}^{(k)}, \frac{x_{\tau-1}^*}{w+1}\right) \right) \\ &= g_{1, T}(x^*; \lambda) + \sum_{k=1}^{w+1} \sum_{\tau \in \Omega_k} d\left(\hat{x}_{\tau-1}^{(k)}, \frac{x_{\tau-1}^*}{w+1}\right). \end{aligned}$$

With these lemmas, we can now prove Theorem 6.

Proof of Theorem 6: Substituting $d(x, y) = \beta \cdot (y - x)^+$ into Lemma 2, we have

$$\begin{aligned} \frac{\text{cost}(SFHC)}{\text{cost}(OPT)} &\leq 1 + \frac{\beta \cdot \sum_{t=1}^T x_{t-1}^*}{(w+1)\text{cost}(OPT)} \\ &\leq 1 + \frac{\beta \cdot \sum_{t=1}^T x_{t-1}^*}{(w+1) \sum_{t=1}^T f_t(x_t^*, \lambda_t)} \quad (15) \\ &\leq 1 + \frac{\beta \cdot \sum_{t=1}^T x_{t-1}^*}{(w+1) f_0 \cdot \sum_{t=1}^T x_t^*} \\ &\leq 1 + \max_s \frac{\beta_s}{(w+1) f_{0,s}} \end{aligned}$$

where the last step uses the facts that $\beta_i/f_{0,i} \leq \max_s \beta_s/f_{0,s}$, and $0 \leq \sum_{t=1}^T x_{t-1}^* \leq \sum_{t=1}^T x_t^*$ elementwise as $x_0^* = 0$. ■

A similar proof establishes Theorem 7.

Proof of Theorem 7:

By the convexity of f ,

$$\begin{aligned} \frac{\text{cost}(AFHC)}{\text{cost}(OPT)} &\leq \frac{1}{w+1} \sum_{k=1}^{w+1} \frac{g_{1, T}(x^{(k)}; \lambda)}{\text{cost}(OPT)} \\ &\leq 1 + \frac{\beta \cdot \sum_{\tau=1}^T x_{\tau-1}^*}{(w+1) \sum_{t=1}^T f_t(x_t^*, \lambda_t)}. \end{aligned}$$

where the second step uses Lemma 1. The result is established by the derivation following (15). ■

C. Proofs of Theorems 1 and 4

We now introduce lemmas used to prove Theorem 4.

We start with two technical lemmas. The first says that the optimal solution on an interval $[i, j]$ is non-decreasing in the initial condition x_{i-1} and the final condition x_{j+1} .

Lemma 3. *Let $S = 1$. Given $\lambda_{i..j} = (\lambda_i, \dots, \lambda_j)$ with each $\lambda_t \in \mathbb{R}^J$, and constants $x_{i-1}, x_{j+1} \in \mathbb{R}$, let $x^{ij} = (x_i^{ij}, \dots, x_j^{ij})$ be a vector minimizing $g_{i,j}(x; \lambda_{i..j}; x_{i-1}) + d(x_j, x_{j+1})$. Then for any $\hat{x}_{i-1} \geq x_{i-1}$ and $\hat{x}_{j+1} \geq x_{j+1}$, there exists a vector $\hat{x}^{ij} = (\hat{x}_i^{ij}, \dots, \hat{x}_j^{ij})$ minimizing $g_{i,j}(x; \lambda_{i..j}; \hat{x}_{i-1}) + d(x_j, \hat{x}_{j+1})$ such that $\hat{x}^{ij} \geq x^{ij}$.*

Proof: Since x^{ij} and \hat{x}^{ij} minimize their respective objectives, we have $g_{i,j}(x^{ij}; \lambda_{i..j}; x_{i-1}) + d(x_j^{ij}, x_{j+1}) \leq g_{i,j}(\hat{x}^{ij}; \lambda_{i..j}; x_{i-1}) + d(\hat{x}_j^{ij}, x_{j+1})$ and $g_{i,j}(\hat{x}^{ij}; \lambda_{i..j}; \hat{x}_{i-1}) + d(\hat{x}_j^{ij}, \hat{x}_{j+1}) \leq g_{i,j}(x^{ij}; \lambda_{i..j}; \hat{x}_{i-1}) + d(x_j^{ij}, \hat{x}_{j+1})$. If there is an x^{ij} such that the latter holds with equality, then we can choose $\hat{x}_i^{ij} = x_i^{ij}$ and consider the problem with $\lambda_{i+1..j}$ recursively. Otherwise, i.e., the latter is a strict inequality, summing the two inequalities and cancelling terms gives

$$(x_i^{ij} - x_{i-1})^+ + (\hat{x}_i^{ij} - \hat{x}_{i-1})^+ + (\hat{x}_{j+1} - x_j^{ij})^+ + (x_{j+1} - x_j^{ij})^+ < (\hat{x}_i^{ij} - x_{i-1})^+ + (x_i^{ij} - \hat{x}_{i-1})^+ + (\hat{x}_{j+1} - x_j^{ij})^+ + (x_{j+1} - \hat{x}_j^{ij})^+.$$

Since $\hat{x}_{i-1} \geq x_{i-1}$ and $\hat{x}_{j+1} \geq x_{j+1}$, it follows that either $x_i^{ij} < \hat{x}_i^{ij}$ or $x_j^{ij} < \hat{x}_j^{ij}$, by the submodularity of $\phi(x, y) = (x - y)^+$. In either case, we can continue recursively, considering $\lambda_{i+1..j}$ in the former case or $\lambda_{i..j-1}$ in the latter.

Finally we have $\hat{x}^{ij} \geq x^{ij}$. \blacksquare

The next technical lemma says that RHC has larger solutions than related algorithms that look less far ahead.

Lemma 4. *Consider a system with homogeneous servers ($S = 1$), and constants $t, X_{t-1} \geq \tilde{x}_{t-1} \geq 0$, and $k \in [t, t + w]$. Let $\tilde{x} = (\tilde{x}_t, \dots, \tilde{x}_{t+k})$ minimize $g_{t,t+k}(x; \lambda, \tilde{x}_{t-1})$, and let $X = x_{RHC}$. Then $\tilde{x} \leq X_{t..t+k}$.*

Proof: The proof is by induction. By hypothesis, $\tilde{x}_{t-1} \leq X_{t-1}$. We need to prove that if $\tilde{x}_{\tau-1} \leq X_{\tau-1}$, then $\tilde{x}_\tau \leq X_\tau$.

Notice that \tilde{x}_τ is the first entry of a vector minimizing $g_{\tau,k}(x, \lambda, x_{\tau-1}) + d(x_k, 0)$ since $d(\cdot, 0) = 0$. Similarly X_τ is the first entry of a vector minimizing $g_{\tau,\tau+w}(x, \lambda, X_{\tau-1})$, which is equivalent to minimizing $g_{\tau,\tau+w}(x, \lambda, X_{\tau-1}) + d(x_{\tau+w}, 0)$. If $k = \tau + w$, we have $\tilde{x}_\tau \leq X_\tau$ by Lemma 3 and the tie-break rule of RHC. Otherwise, i.e., $k < \tau + w$, we know that X_τ is the first entry of a vector minimizing $g_{\tau,k}(x, \lambda, X_{\tau-1}) + d(x_k, x'_{k+1})$ with $x'_{k+1} \geq 0$. By Lemma 3 and the RHC tie-break we again have $\tilde{x}_\tau \leq X_\tau$. \blacksquare

We can now prove the first main lemma used to prove Theorem 4.

Lemma 5. *In a system with homogeneous servers ($S = 1$), each version k of FHC allocates fewer servers than RHC:*

$$x_{FHC}^{(k)} \leq x_{RHC}. \quad (16)$$

Hence $x_{AFHC} \leq x_{RHC}$ and, if f satisfies (10), then $x_{SFHC} \leq x_{RHC}$.

Proof: Let $X = x_{RHC}$ be the result of RHC, and $x = x_{FHC}^{(k)}$. The proof is by induction. By definition, $x_0 = X_0 = 0$. To see that $x_{\tau-1} \leq X_{\tau-1}$ implies $x_\tau \leq X_\tau$, notice that x_τ is the first entry of a vector minimizing $g_{\tau,k}(x, \lambda, x_{\tau-1})$ for some $k \in [\tau, \tau + w]$, with $k + 1 \in \Omega_k$. The implication follows by Lemma 4 and establishes (16). The proof for x_{AFHC} is immediate, and that for x_{SFHC} follows by the scaling of $g_{1,T}$ under (10). \blacksquare

Lemma 6. *If all servers are homogeneous ($S = 1$), then for any given vector $x \leq X$, we have $g_{1,T}(X; \lambda) \leq g_{1,T}(x; \lambda)$.*

Proof: It is sufficient to construct a sequence of vectors ξ^τ such that: $\xi^1 = x$, $\xi_t^\tau = X_t$ for $t < \tau$, and $g_{1,T}(\xi^\tau)$ is non-increasing in τ . The sequence can be constructed inductively with the additional invariant $\xi^\tau \leq X$ as follows.

At stage τ , we calculate $\xi^{\tau+1}$. Apply RHC on $\lambda_\tau, \dots, \lambda_{\tau+w}$ to get $X^\tau(X_{\tau-1}, \lambda) = (\tilde{x}_\tau, \dots, \tilde{x}_{\tau+w})$. Note that $\tilde{x}_\tau = X_\tau \geq \xi_\tau^\tau$ since $\xi^\tau \leq X$ by the inductive hypothesis. Moreover $\tilde{x}_{\tau.. \tau+w} \leq X_{\tau.. \tau+w}$ by Lemma 4.

If $\tilde{x}_{\tau.. \tau+w} \geq \xi_{\tau.. \tau+w}^\tau$ elementwise, then replace elements τ to $\tau + w$ in ξ^τ to get $\xi^{\tau+1} = (\xi_{1.. \tau-1}^\tau, \tilde{x}_{\tau.. \tau+w}, \xi_{\tau+w+1.. T}^\tau) \geq \xi^\tau$. Then $g_{1,\tau+w}(\xi^{\tau+1}; \lambda) \leq g_{1,\tau+w}(\xi^\tau; \lambda)$ by the optimality of $\tilde{x}_{\tau.. \tau+w}$. Since $\xi_{\tau+w}^\tau \leq \tilde{x}_{\tau+w}$ and $d(x, y) = \beta(y - x)^+$ is non-increasing in its first argument, we also have $g_{\tau+w+1,T}(\xi^{\tau+1}; \lambda) \leq g_{\tau+w+1,T}(\xi^\tau; \lambda)$. Therefore, $g_{1,T}(\xi^{\tau+1}; \lambda) \leq g_{1,T}(\xi^\tau; \lambda)$. Finally, to see that $\xi^{\tau+1} \leq X$, note that $\xi^\tau \leq X$ and $\tilde{x}_{\tau.. \tau+w} \leq X_{\tau.. \tau+w}$ as remarked above.

Otherwise, let $k \in [\tau + 1, \tau + w]$ be the minimum index that $\tilde{x}_k < \xi_k^\tau$. Let $\xi^{\tau+1} = (\xi_{1.. \tau-1}^\tau, \tilde{x}_{\tau.. k-1}, \xi_{k.. T}^\tau) \geq \xi^\tau$. Note that $k \geq \tau + 1$ since $\tilde{x}_\tau = X_\tau$; this ensures $\xi_t^\tau = X_t$ for $t < \tau$. Again, $\xi^{\tau+1} \leq X$ as in the previous case. It remains to prove $g_{1,T}(\xi^{\tau+1}; \lambda) \leq g_{1,T}(\xi^\tau; \lambda)$.

Let $u_A = \xi_{\tau.. k-1}^\tau$, $u_B = \xi_{k.. \tau+w}^\tau$, $\tilde{u}_A = \tilde{x}_{\tau.. k-1}$ and $\tilde{u}_B = \tilde{x}_{k.. \tau+w}$. Let vectors (u_A, u_B) , $(\tilde{u}_A, \tilde{u}_B)$, (u_A, \tilde{u}_B) and $(\tilde{u}_A, \tilde{u}_B)$ be indexed by $t \in \{\tau, \dots, \tau + w\}$. To see how replacing $\xi_{\tau.. k-1}^\tau$ by $\tilde{x}_{\tau.. k-1}$ affects the cost in $[1, T]$, note that

$$g_{1,T}(\xi^{\tau+1}; \lambda) - g_{1,T}(\xi^\tau; \lambda) = g_{\tau,\tau+w}((\tilde{u}_A, u_B); \lambda) - g_{\tau,\tau+w}((u_A, u_B); \lambda).$$

Now since $\tilde{x}_k < \xi_k^\tau$, $\tilde{x}_{k-1} \geq \xi_{k-1}^\tau$ and $\phi(x, y) = (x - y)^+$ is submodular, we have

$$\begin{aligned} & (g_{\tau,\tau+w}((\tilde{u}_A, u_B); \lambda) - g_{\tau,\tau+w}((u_A, u_B); \lambda)) \\ & + (g_{\tau,\tau+w}((u_A, \tilde{u}_B); \lambda) - g_{\tau,\tau+w}((\tilde{u}_A, \tilde{u}_B); \lambda)) \quad (17) \\ & = \beta((\xi_k^\tau - \tilde{x}_{k-1})^+ - (\xi_k^\tau - \xi_{k-1}^\tau)^+) \\ & + (\tilde{x}_k - \xi_{k-1}^\tau)^+ - (\tilde{x}_k - \tilde{x}_{k-1})^+ \\ & \leq 0. \end{aligned}$$

But since $(\tilde{u}_A, \tilde{u}_B)$ optimizes (7), we have

$$g_{\tau,\tau+w}((u_A, \tilde{u}_B); \lambda) - g_{\tau,\tau+w}((\tilde{u}_A, \tilde{u}_B); \lambda) \geq 0.$$

Thus the first bracketed term in (17) is non-positive, whence

$$\begin{aligned} & g_{1,T}(\xi^{\tau+1}; \lambda) - g_{1,T}(\xi^\tau; \lambda) \\ & \leq g_{\tau,\tau+w}((\tilde{u}_A, u_B); \lambda) - g_{\tau,\tau+w}((u_A, u_B); \lambda) \\ & \leq 0. \end{aligned}$$

We can now prove that, in the scalar case, RHC outperforms AFHC (and SFHC), and achieves the claimed performance.

Proof of Theorem 4: By Lemma 5 and AFHC, we have $\hat{x} \leq X$. By Lemma 6, $g_{1,T}(X; \lambda) \leq g_{1,T}(\hat{x}; \lambda)$

Proof of Theorem 1: The bound on the competitive ratio of RHC follows from Theorems 4 and 7.

D. “Bad” instances for Receding Horizon Control (RHC)

In this section, we prove the lower bound results in Section III by constructing “bad” instances that force RHC to incur large costs.

Proof of Theorem 2: Consider the operating cost $f_t(x_t, \lambda_t) = f_0 x_t$ for $\lambda_t \leq x_t$ and $f_t(x_t, \lambda_t) = \infty$ for $\lambda_t > x_t$. Note that this cost satisfies both (10) and (11). Now consider the arrival pattern $\lambda = \{\lambda_t\}_{1 \leq t \leq T}$ where $\lambda_{k(w+2)+1} = N$ for $k = 0, 1, \dots$ and other λ_t are all 0. It is easy to see that RHC will give the provisioning $X_{k(w+2)+1} = N$ and $X_t = 0$ for other t . Thus we have

$$g_{1,T}(X; \lambda) = \frac{T}{w+2} N f_0 + \frac{T}{w+2} \beta N.$$

Now consider another provisioning policy $\hat{x} = \{\hat{x}_t = N\}_{1 \leq t \leq T}$. Its cost is $g_{1,T}(\hat{x}; \lambda) = T N f_0 + \beta N$. Thus

$$\begin{aligned} g_{1,T}(X; \lambda) / g_{1,T}(x^*; \lambda) & \geq g_{1,T}(X; \lambda) / g_{1,T}(\hat{x}; \lambda) \\ & = \frac{f_0 + \beta}{(w+2)(f_0 + \beta/T)} \rightarrow \frac{1}{w+2} + \frac{\beta}{(w+2)f_0} \end{aligned}$$

as $T \rightarrow \infty$.

Note that the cost function in the proof of Theorem 2 is applicable to data centers that impose a maximum load on each server (to meet QoS or SLA requirements).

Proof of Theorem 5: When there are $S = 2$ types of server and $J = 1$ type of job, the following instance causes $\text{cost}(SFHC) < \text{cost}(RHC)$.

Choose constants $f_1 > f_2$ and $\beta_1 < \beta_2$ such that $(w+1)f_1 < (w+1)f_2 + \beta_2 < (w+1)f_1 + \beta_1$ and $wf_1 + \beta_1 < wf_2 + \beta_2$. These can simultaneously be achieved by choosing an arbitrary $f_1 - f_2 > 0$, then choosing $\beta_2 - \beta_1 \in (w, w+1)(f_1 - f_2)$, and then $\beta_2 > (w+1)(f_1 - f_2)$.

Let the switching cost for a type i server be β_i . Let the operating cost be $f_t(x_t, \lambda_t) = f_1 x_{t,1} + f_2 x_{t,2}$ for $\lambda_t \leq x_{t,1} + x_{t,2}$ and $f_t(x_t, \lambda_t) = \infty$ for $\lambda_t > x_{t,1} + x_{t,2}$. Note that this satisfies both (10) and (11). In this system, type-2 servers have lower operating cost but higher switching cost.

Choose constants $T > w + \max(1, \beta_2/(f_1 - f_2))$, and $N > 0$. Now consider the cost of schemes AFHC, SFHC and RHC under the load such that: (a) $\lambda_{w+1} = 0$, (b) $\lambda_t = N$ for all other $t \in [1, T]$, and (c) $\lambda_t = 0$ for $t \notin [1, T]$.

Under SFHC: At time $t = 1$, group 1 sees $\lambda_1, \dots, \lambda_{w+1}$ and so uses $N/(w+1)$ type-1 servers for the first w timeslots and

turns off all servers at timeslot $w+1$. From timeslot $t = w+2$ onwards, it sees $\lambda = N$ until time T , and so uses $N/(w+1)$ type-2 servers until T .

For $2 \leq i \leq w+1$, group i initially sees a window of loads in which w or fewer time slots have non-zero load, and so again chooses servers of type 1. However, the last slot in the first window, slot $i-1$, has load N , and so all servers remain on. In the second and subsequent windows, the cost of switching is greater than uses type-1 servers until T . Thus its total cost is

$$\begin{aligned} \text{cost}(SFHC) & = \frac{w}{w+1} (f_1 N T + \beta_1 N) \\ & + \frac{1}{w+1} (f_1 N w + \beta_1 N + f_2 N (T - w - 1) + \beta_2 N) \end{aligned}$$

The unique feasible dispatching is for each group to send all N/w traffic to the N/w servers that are on.

Under AFHC: the provisioning is the same as SFHC, and the dispatching is the same unique feasible one.

Under RHC: X uses type-1 servers forever, for the same reason as group $i \geq 2$ under SFHC. Thus its total cost is

$$\text{cost}(RHC) = f_1 N T + \beta_1 N$$

The choice of T implies $f_1(T-w) > f_2(T-w-1) + \beta_2$, and thus $\text{cost}(SFHC) < \text{cost}(RHC)$.

Proof of Theorem 3: The proof will be by construction. Let $J \geq S \gg w$. Let the switching cost for a type- s server be $\beta_s = \beta_0 + 2\epsilon s w$. Denote the type- j workload at time t by $\lambda_{t,j}$ ($j \in \{1, \dots, J\}$). Let the operating cost be $f(x_t, \lambda_t) = \sum_{s=1}^S (f_{0,0} - s\epsilon + C \sum_{l=s+1}^J \lambda_{t,l}) x_{t,s}$ for $\sum_{s=1}^S x_{t,s} \geq \sum_{j=1}^J \lambda_{t,j}$ and $f(x_t, \lambda_t) = \infty$ otherwise, where $\epsilon > 0$ is a small constant and $C > \max_s \beta_s$ is a large constant. Intuitively, this operating cost function means that lower types of servers consume a little bit more energy, but type- s servers are very inefficient at processing workload of types higher than s . Also, the switching cost increases slightly with the type of the server. This may occur if all servers use roughly the same hardware, but servers of type s store locally only data for jobs of types 1 to s .

Consider the workload trace which has $\lambda_{t,1} = N$ for $t = 1, \dots, w+1$ and $\lambda_{t,t-w} = N$ for $t = w+2, \dots, w+S$. All the other arrival rates $\lambda_{t,j}$ are zero. Then RHC would start with N type-1 servers (the cheapest to turn on) at timeslot 1, and then at each $t \in [2, S]$ would switch off the type- $(t-1)$ servers and turn on N type- t servers (the cheapest way to avoid the excessive cost of processing type t jobs using servers of type $< t$). For sufficiently small ϵ , the optimal solution always uses N type- S servers for $t \in [1, w+S]$. Therefore the total costs in $[1, w+S]$ for small ϵ are $\text{cost}(RHC) = N(w+S)f_0 + N S \beta_0 + O(\epsilon)$ and $\text{cost}(OPT) = N(w+S)f_0 + N \beta_0 + O(\epsilon)$. Therefore,

$$\frac{\text{cost}(RHC)}{\text{cost}(OPT)} = 1 + \frac{(S-1)\beta_0}{(w+S)f_0 + \beta_0} + O(\epsilon).$$

For $S \gg w$ and $S f_0 \gg \beta_0$ and small ϵ , this ratio will approach $1 + \beta_0/f_0$, which implies the result.

V. CONCLUDING REMARKS

This paper provides new analytic results for a classic algorithm, Receding Horizon Control (RHC). We show that, in the context of dynamic capacity provisioning, RHC is $1 + O(1/w)$ -competitive when the system is homogeneous, but that when the system is heterogeneous RHC can perform badly — the competitive ratio does not improve as the size of the prediction window, w , grows. Accordingly, we develop two new algorithms that use a mixture or average of $w + 1$ Fixed Horizon Control (FHC) algorithms. These algorithms are able to provide $1 + O(1/w)$ -competitive ratios even in the heterogeneous setting.

A key feature of this paper is the generality of the model considered and the analytical results proven. The model captures a wide variety of dynamic capacity provisioning problems (e.g., geographical load balancing and provisioning within a heterogeneous data center), and the analytical results hold for quite general cost functions (typically not even requiring convexity, as is typically assumed).

This work is only a first step toward developing analytical foundations for dynamic capacity provisioning problems in heterogeneous systems. There are many questions that this paper leaves open. For example, we have not proven lower bounds for the AFHC and SFHC algorithms introduced here. More generally, it is not clear whether other algorithms could provide better competitive ratios in the heterogeneous setting as there is no proven lower bound on the optimal competitive ratio achievable by an online algorithm in this setting. Additionally, it would be interesting to provide average-case (rather than worst-case) results for the algorithms discussed here.

ACKNOWLEDGEMENTS

This work was supported by NSF grants CCF 0830511, CNS 0911041, and CNS 0846025 MURI grant W911NF-08-1-0233, Microsoft Research, the Lee Center for Advanced Networking, and ARC grant FT0991594. We would also like thank Zhenhua Liu and Steven Low for the useful feedback they have provided on this work.

REFERENCES

- [1] N. Bansal, H.-L. Chan, and K. Pruhs, “Speed scaling with an arbitrary power function,” in *Proc. ACM-SIAM Symp. Discrete Algorithms (SODA)*, 2009, pp. 693–701.
- [2] A. Wierman, L. L. H. Andrew, and A. Tang, “Power-aware speed scaling in processor sharing systems,” in *Proc. IEEE INFOCOM*, 2009, pp. 2007–2015.
- [3] L. L. H. Andrew, M. Lin, and A. Wierman, “Optimality, fairness and robustness in speed scaling designs,” in *Proc. ACM SIGMETRICS*, 2010, pp. 37–48.
- [4] Z. Liu, M. Lin, A. Wierman, S. H. Low, and L. L. H. Andrew, “Greening geographical load balancing,” in *Proc. ACM SIGMETRICS*, San Jose, CA, 7-11 Jun 2011, pp. 233–244.
- [5] A. Gandhi, V. Gupta, M. Harchol-Balter, and M. Kozuch, “Optimality analysis of energy-performance trade-off for server farm management,” *Performance Evaluation*, no. 11, pp. 1155–1171, Nov. 2010.
- [6] M. Lin, A. Wierman, L. L. H. Andrew, and E. Thereska, “Dynamic right-sizing for power-proportional data centers,” in *Proc. IEEE INFOCOM*, 2011, pp. 1098–1106.

- [7] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle, “Managing energy and server resources in hosting centers,” in *Proc. ACM Symp. Operating System Principles (SOSP)*, 2001, pp. 103–116.
- [8] X. Wang and M. Chen, “Cluster-level feedback power control for performance optimization,” in *IEEE Int. Symp. High Performance Computer Architecture (HPCA)*, 2008, pp. 101–110.
- [9] D. Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy, and G. Jiang, “Power and performance management of virtualized computing environments via lookahead control,” *Cluster computing*, vol. 12, no. 1, pp. 1–15, Mar. 2009.
- [10] W. Xu, X. Zhu, S. Singhal, and Z. Wang, “Predictive control for dynamic resource allocation in enterprise data centers,” in *Proc. IEEE/IFIP Netw. Op. Manag. Symp (NOMS)*, 2006, pp. 115–126.
- [11] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper, “Workload analysis and demand prediction of enterprise data center applications,” in *Proc. IEEE Symp. Workload Characterization (IISWC)*, Boston, MA, Sep. 2007, pp. 171–180.
- [12] W. Kwon and A. Pearson, “A modified quadratic cost problem and feedback stabilization of a linear system,” *IEEE Trans. Automatic Control*, vol. AC-22, no. 5, pp. 838–842, 1977.
- [13] W. H. Kwon, A. M. Bruckstein, and T. Kailath, “Stabilizing state feedback design via the moving horizon method,” *Int. J. Contr.*, vol. 37, no. 3, pp. 631–643, 1983.
- [14] D. Q. Mayne and H. Michalska, “Receding horizon control of nonlinear systems,” *IEEE Trans. Automat. Contr.*, vol. 35, no. 7, pp. 814–824, 1990.
- [15] N. Buchbinder, N. Jain, and I. Menache, “Online job-migration for reducing the electricity bill in the cloud,” in *Proc. Networking 2011*. Springer, 2011, pp. 172–185.
- [16] A. Krioukov, P. Mohan, S. Alspaugh, L. Keys, D. Culler, and R. Katz, “Napsac: design and implementation of a power-proportional web cluster,” *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 1, pp. 102–108, 2011.
- [17] E. Pinheiro, R. Bianchini, E. Carrera, and T. Heath, “Dynamic cluster reconfiguration for power and performance,” in *Compilers and Operating Systems for Low Power*, L. Benini, M. Kandemir, and J. Ramanujam, Eds. Kluwer Academic Publishers, 2002.
- [18] T. Heath, B. Diniz, E. Carrera, W. Meira Jr, and R. Bianchini, “Energy conservation in heterogeneous server clusters,” in *Proc. ACM SIGPLAN Symp. Princip. prac. parall. prog.*, 2005, pp. 186–195.
- [19] T. Horvath and K. Skadron, “Multi-mode energy management for multi-tier server clusters,” in *Proc. ACM Int. Conf. Parallel Architectures and Compilation Techniques (PACT)*, 2008, p. 1.
- [20] A. Borodin, N. Linial, and M. Saks, “An optimal on-line algorithm for metrical task system,” *Journal of the ACM*, vol. 39, no. 4, pp. 745–763, 1992.
- [21] D. Sleator and R. Tarjan, “Amortized efficiency of list update and paging rules,” *Communications of the ACM*, vol. 28, no. 2, pp. 202–208, 1985.
- [22] M. Manasse, L. McGeoch, and D. Sleator, “Competitive algorithms for on-line problems,” in *Proc. ACM symposium on Theory of computing (STOC)*, 1988, pp. 322–333.
- [23] S. Albers, “On the influence of lookahead in competitive paging algorithms,” *Algorithmica*, vol. 18, no. 3, pp. 283–305, 1997.
- [24] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs, “Cutting the electric bill for internet-scale systems,” in *ACM SIGCOMM*, Aug. 2009, pp. 123–134.
- [25] A. Mohsenian-Rad and A. Leon-Garcia, “Coordination of cloud computing and smart power grids,” in *Proc. IEEE Smart Grid Communications (SmartGridComm)*, 2010, pp. 368–372.
- [26] H. Hofstee, “Power efficient processor architecture and the cell processor,” in *Int. Symp. High-Perf. Comp. Arch. (HPCA)*, 2005, pp. 258–262.
- [27] E. Koutsoupias and C. Papadimitriou, “Beyond competitive analysis,” in *SIAM Journal on Computing*, 2000.
- [28] J. Augustine, S. Irani, and C. Swamy, “Optimal power-down strategies,” *SIAM Journal on Computing*, vol. 37, no. 5, pp. 1499–1516, 2008.
- [29] E. Koutsoupias, “The k -server problem,” *Computer Science Review*, vol. 3, no. 2, pp. 105–118, 2009.