Dynamic right-sizing for power-proportional data centers — Extended version

Minghong Lin, Adam Wierman, Lachlan L. H. Andrew and Eno Thereska

***Abstract*—Power consumption imposes a significant cost for data centers implementing cloud services, yet much of that power is used to maintain excess service capacity during periods of low load. This paper investigates how much can be saved by dynamically "right-sizing" the data center by turning off servers during such periods, and how to achieve that saving via an online algorithm. We propose a very general model and prove that the optimal offline algorithm for dynamic right-sizing has a simple structure when viewed in reverse time, and this structure is exploited to develop a new "lazy" online algorithm, which is proven to be 3-competitive. We validate the algorithm using traces from two real data center workloads and show that significant cost-savings are possible. Additionally, we contrast this new algorithm with the more traditional approach of receding horizon control.**

I. Introduction

Energy costs represent a significant fraction of a data center's budget [1] and this fraction is expected to grow as the price of energy increases in coming years. Hence, there is a growing push to improve the energy efficiency of the data centers behind cloud computing. A guiding focus for research into "green" data centers is the goal of designing data centers that are "power-proportional", i.e., use power only in proportion to the load. However, current data centers are far from this goal – even today's energy-efficient data centers consume almost half of their peak power when nearly idle [2].

A promising approach for making data centers more power-proportional is using software to dynamically adapt the number of active servers to match the current workload, i.e., to dynamically 'right-size" the data center. Specifically, dynamic right-sizing refers to adapting the way requests are dispatched to servers in the data center so that, during periods of low load, servers that are not needed do not have jobs routed to them and thus are allowed to enter a power-saving mode (e.g., go to sleep or shut down).

Technologies that implement dynamic right-sizing are still far from standard in data centers due to a number of challenges. First, servers must be able to seamlessly transition into and out of power saving modes while not losing their state. There has been a growing amount of research into enabling this in recent years, dealing with virtual machine state [3], network state [4] and storage state [5], [6]. Second, such techniques must prove to be reliable, since administrators we talk to worry about wear-and-tear consequences of such technologies. Third, and the challenge that this paper addresses, it is unclear how to determine the number of servers to toggle into power-saving mode and how to control servers and requests 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. Although Receding Horizon Control has been proposed for the online control [7], [8], as shown in Section III, its performance really depends on the prediction window and the toggling cost of servers, which may vary widely in different data centers.

A goal of this paper is to provide a new algorithm to address this challenge for general settings. To this end, we develop a general model that captures the major issues that affect the design of a right-sizing algorithm, including: the cost (lost revenue) associated with the increased delay from using fewer servers, the energy cost of maintaining an active server with a particular load, and the cost incurred from toggling a server into and out of a power-saving mode (including the delay, energy, and wear-and-tear costs).

This paper makes three contributions: First, we analytically characterize the optimal offline solution (Section IV). We prove that it exhibits a simple, "lazy" structure when viewed in reverse time.

Second, we introduce and analyze a novel, practical online algorithm motivated by this structure (Section V). The algorithm, named *Lazy Capacity Provisioning (LCP(w))*, uses a prediction window of length w of future arrivals and mimics the "lazy' structure of the optimal algorithm, but proceeding forward instead of backwards in time. We prove that LCP(w) is 3-competitive, i.e., its cost is at most 3 times that of the optimal offline solution. This is regardless of the workload and for very general energy and delay cost models, even when no information is used about arrivals beyond the current time period ($w = 0$). Further, in practice, LCP(w) is far better than 3-competitive, incurring nearly the optimal cost.

Third, we validate our algorithm using two load traces (from Hotmail and a Microsoft Research data center) to evaluate the cost savings achieved from dynamic right-sizing in practice (Section VI). We contrast our new algorithm with Receding Horizon Control and confirm that our algorithm provides much more stable cost saving. We show that significant savings are possible under a wide range of settings and that savings become dramatic when the workload is predictable over an interval proportional to the toggling cost. The magnitude of the potential savings depends primarily on the peak-to-mean ratio (PMR) of the workload, with a PMR of 3 being enough to give 30% cost saving. In the context of these real traces, we also discuss when it does and when it does not make sense to use dynamic right-sizing versus the alternative of "valley-filling", i.e., using periods of low load to run background/ maintenance tasks. We find that dynamic right-sizing provides

M. Lin and A. Wierman are with California Institute of Technology.
L. L. H. Andrew is with Swinburne University of Technology.
E. Thereska is with Microsoft Research, Cambridge.

more than 15% cost savings even when the background work makes up 40% of the workload when the PMR is larger than 3.

II. Model formulation

Right-sizing problems vary between data centers: some systems have flexible quality of service (QoS) requirement, others have hard service level agreements (SLAs); the electricity price may be time-varying or fixed; workloads can be homogeneous or heterogeneous, etc. We first introduce a general model which captures the main issues in many right-sizing problems in different systems. We then give examples and show how they fit into this general model.

A. General model

We now describe the general model used to explore the cost savings possible from dynamic right-sizing. An instance consists of a constant $\beta > 0$, a horizon[1] $T > 0$, a sequence of non-negative convex functions $g_t(\cdot)$ for $t = 1, \ldots, T-1$; $g_t(\cdot)$ can take on the value ∞ but cannot be identically ∞. For notational simplicity, let $g_T(x) \equiv 0$ and $x_0 = x_T = 0$. The model is:

$$\begin{aligned} \underset{x_1,\ldots,x_{T-1}}{\text{minimize}} \quad & \sum_{t=1}^{T} g_t(x_t) + \beta \sum_{t=1}^{T} (x_t - x_{t-1})^+ \qquad (1) \\ \text{subject to} \quad & x_t \geq 0 \end{aligned}$$

where $(x)^+ = \max(0, x)$, and $\{x_t\}$ are non-negative scalar variables. The solution to optimization (1) may not be unique. By Corollary 1 in Appendix A, there exists a solution that is the elementwise maximum. Unless otherwise stated, "the solution" refers to this maximum solution. We discuss the extension to vector x_t in Section VIII.

To apply this to data center right-sizing, recall the major costs of right-sizing: (a) the cost associated with the increased delay from using fewer servers and the energy cost of maintaining an active server with a particular load; (b) the cost incurred from toggling a server into and out of a power-saving mode (including the delay, migration, and wear-and-tear costs). We call the first part "operating cost" and the second part "switching cost" and consider a discrete-time model where the timeslot length matches the timescale at which the data center can adjust its capacity. There is a (possibly long) time-interval of interest $t \in \{0, 1, \ldots, T\}$ and the capacity (i.e., number of active servers) at time t is x_t.

The operating cost in each timeslot is modelled by $g_t(x_t)$, which presents the cost of using x_t servers (x_t has a vector value if servers are heterogeneous) to serve requests at timeslot t. Moreover, $g_t(\cdot)$ captures many other factors in timeslot t, including the arrival rate, the electricity price, the cap on available servers, the SLA constraints.

For the switching cost, let β be the cost to transition a server from the sleep state to the active state and back again. We deem this cost to be incurred only when the server wakes up. Thus the switching cost for changing the number of active servers from x_{t-1} to x_t is $\beta(x_t - x_{t-1})^+$. The constant β includes the costs of (i) the energy used, (ii) the delay in migrating connections/data/etc. (e.g., by VM techniques), (iii) increased wear-and-tear on the servers, and (iv) the risk associated with server toggling. If only (i) and (ii) matter, then β is either on the order of the cost to run a server for a few seconds (waking from suspend-to-RAM or migrating network state [4] or storage state [5]), or several minutes (to migrate a large VM [3]). However, if (iii) is included, then β becomes on the order of the cost to run a server for an hour [9]. Finally, if (iv) is considered then our conversations with operators suggest that their perceived risk that servers will not turn on properly when toggled is high, so β may be many hours' server costs. Throughout, denote the operating cost of a vector $X = (x_1, \ldots, x_T)$ by $cost_o(X) = \sum_{t=1}^{T} g_t(x_t)$, the switching cost by $cost_s(X) = \beta \sum_{t=1}^{T} (x_t - x_{t-1})^+$, and $cost(X) = cost_o(X) + cost_s(X)$.

We have seen that Formulation (1) captures the operating cost and switching cost of right-sizing problems. However, Formulation (1) may look not that general at first glance due to the lack of constraints. Actually constraints in right-sizing problems can be handled implicitly in functions $g_t(\cdot)$ by *extended-value extension*, i.e., defining $g_t(\cdot)$ to be ∞ outside its domain. This extension makes our description/ notation more clear without introducing particular constraints in different data centers.

Formulation (1) makes a simplification that it does not enforce that x_t be integer valued. This is acceptable since the number of servers in a typical data center is large; moreover, preliminary investigation suggests that this can be enforced by an appropriate choice of cost functions, $g_t(\cdot)$. This model also ignores issues surrounding reliability and availability. In practice, a solution that toggles servers must still maintain the reliability and availability guarantees. Such issues are beyond the scope of this paper, however previous work shows that solutions are possible [5].

This optimization problem would be easy to solve *offline*, i.e., given functions $g_t(\cdot)$ for all t. However, our goal is to find *online* algorithms for this optimization, i.e., algorithms that determine x_τ using only information up to time $\tau + w$, where the "prediction window" $w \geq 0$ is part of the problem instance. Here, we assume that the predictions $[g_\tau, \ldots, g_{\tau+w}]$ are known perfectly at time τ, but we show in Section VI that our algorithm is robust to this assumption in practice.

We evaluate the performance of an online algorithm $\mathcal{A}$ using the standard notion of *competitive ratio*. The competitive ratio of $\mathcal{A}$ is defined as the supremum, taken over all possible inputs, of $cost(\mathcal{A})/cost(OPT)$, where $cost(\mathcal{A})$ is the objective function of (1) under $\mathcal{A}$ and OPT is the optimal offline algorithm. The analytic results of Sections IV and V assume that the service has a finite duration, i.e. $T < \infty$, but hold for arbitrary sequences of convex functions $g_t(\cdot)$. Thus, the analytic results provide worst-case guarantees. However, to provide realistic cost estimates, we consider case-studies in Section VI where $g_t(\cdot)$ is based on real-world traces.

[1] If parameters such as β are known in advance, then the results in this paper can be extended to a model in which each instance has a finite duration, but the cost is summed over an infinite horizon.

B. Special cases

Now let us consider two choices of $g_t(\cdot)$ to fit two different right-sizing problems into Formulation (1). The discrete-time model consists of finitely many times $t \in \{0, 1, \ldots, T\}$, where T may be a year, measured in timeslots of 10 minutes. Assume the workload affects the operating cost at timeslot t only by its mean arrival rate, denoted λ_t. (This may have a vector value if there are multiple types of work). This assumption is reasonable for many services such as web search, social network or email service since the request interarrival times and the response times are much shorter than the timeslots so that the provisioning can be based on the arrival rate.

We model a data center as a collection of homogeneous servers (except Section VIII where heterogeneous systems are discussed) and focus on determining x_t, the number of active servers during each time slot t. For notational simplicity, we assume a load balancer assigns arriving jobs to active servers uniformly, at rate λ_t/x_t per server (which is widely deployed and turns out to be optimal for many systems). Assume the power of an active server handling arrival rate λ is $e(\lambda)$, and the performance metric we care about is $d(\lambda)$ (e.g., average response time or response time violation probability). For example, a common model of the power for typical servers is an affine function $e(\lambda) = e_0 + e_1\lambda$ where e_0 and e_1 are constants; e.g., see [10]. The average response time can be modeled using standard queuing theory results. If the server happens to be modeled by an M/GI/1 Processor Sharing queue then the average response time is $d(\lambda) = 1/(\mu-\lambda)$, where the service rate of the server is μ [11]. Other examples include, for instance, using the 99th percentile of delay instead of the mean. In fact, if the server happens to be modeled by an M/M/1 Processor Sharing queue then the 99th percentile is $\log(100)/(\mu - \lambda)$, and so the form of $d(\lambda)$ does not change [11]. Note that, in practice, $e(\lambda)$ and $d(\lambda)$ can be empirically measured by observing the system over time without assuming mathematical models for them. Our analytical results work for any $e(\cdot)$ and $d(\cdot)$ as long as they result in convex $g_t(\cdot)$. But for simplicity, we will use $e(\lambda) = e_0+e_1\lambda$ and $d(\lambda) = 1/(\mu-\lambda)$ for our numerical experiments.

1) Example 1: Services with flexible QoS requirement: Many Internet services, such as web search, email service and social network are flexible in response time in certain range. For such applications, it is hard to find a threshold to distinguish "good services" and "bad services". Instead, the revenue is lost gradually as the response time increases and thus our goal is to balance the performance and the power cost [8], [12]–[14]. One natural model for the lost revenue is $d_1\lambda(d(\lambda) - d_0)^+$ where d_0 is the minimum delay users can detect and d_1 is a constant. This measures the perceived delay weighted by the fraction of users experiencing that delay. For the energy cost, assume the electricity price is p_t at timeslot t, then the energy cost is $p_t e(\lambda)$. There may be a time-varying cap M_t on the number of active server available due to other activities/services in the data center (e.g., backup activities). The combination together with the dispatching rule at the load balancer gives

$$\begin{aligned} \text{minimize } & \sum_{t=1}^{T} x_t \left(\frac{d_1\lambda_t}{x_t} \left(d\left(\frac{\lambda_t}{x_t}\right) - d_0 \right)^+ + p_t e\left(\frac{\lambda_t}{x_t}\right) \right) \\ & + \beta \sum_{t=1}^{T} (x_t - x_{t-1})^+ \qquad (2) \\ \text{subject to } & \lambda_t \le x_t \le M_t \end{aligned}$$

Compared to Formulation (1), we have $g_t(x_t) =$

$$\begin{cases} x_t \left(\frac{d_1\lambda_t}{x_t} \left(d\left(\frac{\lambda_t}{x_t}\right) - d_0 \right)^+ + p_t e\left(\frac{\lambda_t}{x_t}\right) \right) & \text{if } \lambda_t \le x_t \le M_t, \\ \infty & \text{otherwise.} \end{cases}$$

We can see that $g_t(x_t)$ is in the form of the perspective function of $f(z) = d_1 z(d(z) - d_0)^+ + p_t e(z)$, which is convex under common models of $d(\cdot)$ and $e(\cdot)$, such as M/G/1 queueing models with or without speed scaling [15] of individual servers. Thus $g_t(\cdot)$ is convex under common models.

2) Example 2: Services with hard QoS constraints: Some data centers have to support services with hard constraints such as SLA requirement or delay constraints for multimedia applications. For such systems, the goal is to minimize energy cost while satisfy the constraints, as shown in [16], [17]. The optimization becomes

$$\begin{aligned} \text{minimize } & \sum_{t=1}^{T} x_t p_t e\left(\frac{\lambda_t}{x_t}\right) + \beta \sum_{t=1}^{T} (x_t - x_{t-1})^+ \qquad (3) \\ \text{subject to } & x_t \ge \lambda_t \text{ and } d\left(\frac{\lambda_t}{x_t}\right) \le D_t \end{aligned}$$

where $d(\cdot)$ can be average delay, delay violation probability or other performance metrics. This constraint usually defines a convex domain for x_t. Actually if we assume that $d(\cdot)$ is an increasing function of the load, then it is equivalent to imposing an upper bound on the load per server (i.e., λ_t/x_t is less than certain threshold) and thus results in a convex optimization problem.

The corresponding $g_t(\cdot)$ can be defined as follows:

$$g_t(x_t) = \begin{cases} x_t p_t e\left(\frac{\lambda_t}{x_t}\right) & \text{if } x_t \ge \lambda_t \text{ and } d\left(\frac{\lambda_t}{x_t}\right) \le D_t, \\ \infty & \text{otherwise.} \end{cases}$$

III. Receding Horizon Control

As motivation for deriving a new algorithm, let us first consider the standard approach. An online policy that is commonly proposed to control data centers [7], [8] (and other dynamic systems) is Receding Horizon Control (RHC) algorithms, also known as Model Predictive Control. RHC has a long history in the control theory literature [18]–[20] where the focus was on stability analysis. In this section, let us introduce briefly the competitive analysis result of RHC for our data center problem (which is proven in [21]) and see why we may need a better online algorithm.

Informally, RHC(w) works by solving, at time τ, the cost optimization over the window $(\tau, \tau+w)$ given the starting state

$x_{\tau-1}$, and then using the first step of the solution, discarding the rest.

Formally, define $X^\tau(x_{\tau-1}; g_\tau \dots g_{\tau+w})$ as the vector in $\mathbb{R}^{w+1}$ indexed by $t \in \{\tau, \dots, \tau+w\}$, which is the solution to

$$\begin{aligned} \text{minimize} \quad & \sum_{t=\tau}^{\tau+w} g_t(x_t) + \beta \sum_{t=\tau}^{\tau+w} (x_t - x_{t-1})^+ \\ \text{subject to} \quad & x_t \geq 0 \end{aligned} \tag{4}$$

Then, RHC(w) works as follows.

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

$$x_\tau = X^\tau_\tau(x_{\tau-1}; g_\tau \dots g_{\tau+w})$$

Note that (4) need not have a unique solution. We define RHC(w) to select the solution with the greatest first entry. Define e_0 the minimum cost per timeslot for an active server, then we have the following theorem [21]:

Theorem 1. *RHC(w) is $(1 + \frac{\beta}{(w+1)e_0})$-competitive for optimization (1) but not better than $(\frac{1}{w+2} + \frac{\beta}{(w+2)e_0})$-competitive.*

This highlights that, with enough lookahead, RHC(w) is guaranteed to perform quite well. However, its competitive ratio depends on the parameters β, e_0 and w . These parameters vary widely in different data centers since they host different services or have different SLAs. Thus, RHC(w) may have a poor competitive ratio for a data center with big switching cost or small prediction window.

This poor performance is to be expected since RHC(w) makes no use of the structure of the optimization we are trying to solve. We may wonder if there exists better online algorithm for this problem that performs well for a wider range of parameters, even when the prediction window $w = 0$ (no workload prediction except current timeslot).

IV. The optimal offline solution

In order to exploit the structure of the specific problem (1), the first natural task is to characterize the optimal offline solution, i.e., the optimal solution given $g_t(\cdot)$ for all t. The insight provided by the characterization of the offline optimum motivates the formulation of our online algorithm.

It turns out that there is a simple characterization of the optimal offline solution X^* to the optimization problem (1), in terms of two bounds on the optimal solution which correspond to charging cost β either when a server comes out of power-saving mode (as (1) states) or when it goes in. The optimal x^*_τ can be viewed as "lazily" staying within these bounds going backwards in time.

More formally, let us first describe lower and upper bounds on x^*_τ, denoted x^L_τ and x^U_τ, respectively. Let $(x^L_{\tau,1}, \dots, x^L_{\tau,\tau})$ be the solution vector to the optimization problem

$$\begin{aligned} \text{minimize} \quad & \sum_{t=1}^{\tau} g_t(x_t) + \beta \sum_{t=1}^{\tau} (x_t - x_{t-1})^+ \\ \text{subject to} \quad & x_t \geq 0 \end{aligned} \tag{5}$$

where $x_0 = 0$. Then, define $x^L_\tau = x^L_{\tau,\tau}$. Similarly, let $(x^U_{\tau,1}, \dots, x^U_{\tau,\tau})$ be the solution vector to the optimization

$$\begin{aligned} \text{minimize} \quad & \sum_{t=1}^{\tau} g_t(x_t) + \beta \sum_{t=1}^{\tau} (x_{t-1} - x_t)^+ \\ \text{subject to} \quad & x_t \geq 0 \end{aligned} \tag{6}$$

Then, define $x^U_\tau = x^U_{\tau,\tau}$.

Notice that in each case, the optimization problem includes only times $1 \leq t \leq \tau$, and so ignores the future information for $t > \tau$. In the case of the lower bound, β cost is incurred for each server toggled on, while in the upper bound, β cost is incurred for each server toggled into power-saving mode.

Lemma 1. *For all τ, $x^L_\tau \leq x^*_\tau \leq x^U_\tau$.*

Given Lemma 1, we now characterize the optimal solution x^*_τ. Define $(x)^b_a = \max(\min(x, b), a)$ as the projection of x into $[a, b]$. Then, we have:

Theorem 2. *The optimal solution $X^* = (x^*_1, \dots, x^*_T)$ of the data center optimization problem (1) satisfies the backward recurrence relation*

$$x^*_\tau = \begin{cases} 0, & \tau > T; \\ \left(x^*_{\tau+1}\right)^{x^U_\tau}_{x^L_\tau}, & \tau \leq T. \end{cases} \tag{7}$$

Theorem 2 and Lemma 1 are proven in Appendix A.

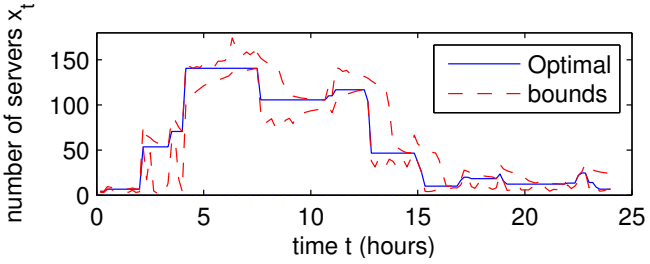
An example of the optimal x^*_t can be seen in Figure 1(a). Many more numeric examples of the performance of the optimal offline algorithm are provided in Section VI.

Theorem 2 and Figure 1(a) highlight that the optimal algorithm can be interpreted as moving backwards in time, starting with $x^*_T = 0$ and keeping $x^*_\tau = x^*_{\tau+1}$ unless the bounds prohibit this, in which case it makes the smallest possible change. This interpretation hightlights that it is impossible for an online algorithm to compute x^*_τ since, without knowledge of the future, an online algorithm cannot know whether to keep x_τ constant or to follow the upper/lower bound.

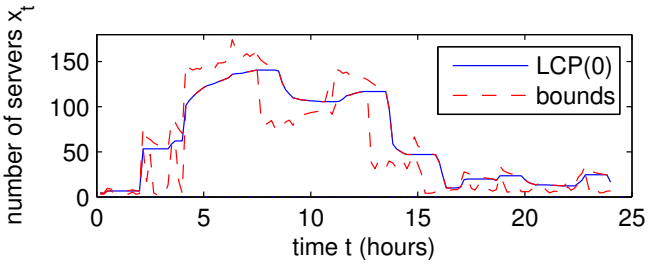
V. Lazy capacity provisioning

A major contribution of this paper is the presentation and analysis of a novel *online* algorithm, Lazy Capacity Provisioning (LCP(w)). At time τ, LCP(w) knows only $g_t(\cdot)$ for $t \leq \tau + w$, for some prediction window w. As mentioned before, we assume that these are known perfectly, but we show in Section VI that the algorithm is robust to this assumption in practice. The design of LCP(w) is motivated by the structure of the optimal offline solution described in Section IV. Like the optimal solution, it "lazily" stays within upper and lower bounds. However, it does this moving forward in time instead of backwards in time.

Before defining LCP(w) formally, recall that the bounds x^U_τ and x^L_τ do not use knowledge about the loads in the prediction window of LCP(w). To use it, define refined bounds $x^{U,w}_\tau$ and $x^{L,w}_\tau$ such that $x^{U,w}_\tau = x^U_{\tau+w,\tau}$ in the solution of (6) and $x^{L,w}_\tau = x^L_{\tau+w,\tau}$ in that of (5). Note that $w = 0$ is allowed (no future prediction) and $x^{U,0}_\tau = x^U_\tau$ and $x^{L,0}_\tau = x^L_\tau$. The following generalization of Lemma 1 is proven in Appendix B.

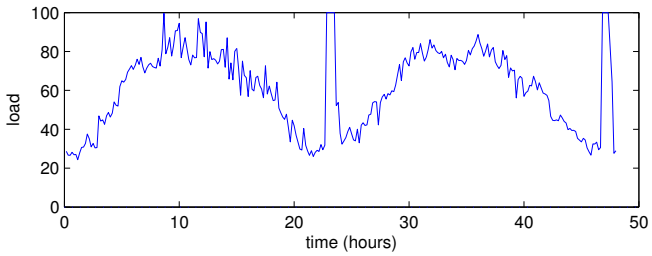


(a) Offline optimal

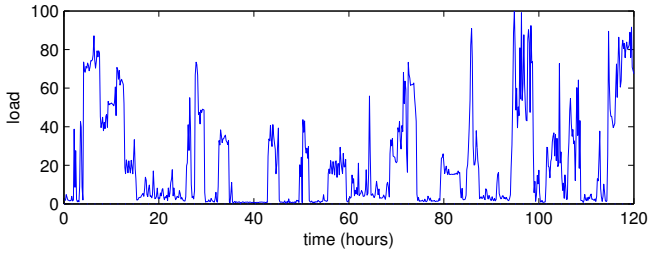


(b) LCP(0)

Fig. 1. Illustrations of (a) the offline optimal solution and (b) LCP(0) for the first day of the MSR workload described in Section VI with a sampling period of 10 minutes. The operating cost is defined in (2) with $d_0 = 1.5$, $d_1 = 1$, $\mu = 1$, $p_t = 1$, $e_0 = 1$ and $e_1 = 0$ and the switching cost has $\beta = 6$ (corresponding to the energy consumption for one hour).



(a) Hotmail



(b) MSR

Fig. 2. Illustration of the traces used for numerical experiments.

Lemma 2. *$x_\tau^L \le x_\tau^{L,w} \le x_\tau^* \le x_\tau^{U,w} \le x_\tau^U$ for all $w \ge 0$.*

Now, we are ready to define LCP(w) using $x_\tau^{U,w}$ and $x_\tau^{L,w}$.

Algorithm 2. ***Lazy Capacity Provisioning, LCP(w).*** *Let $X^{LCP(w)} = (x_0^{LCP(w)}, \dots, x_T^{LCP(w)})$ denote the vector of active servers under LCP(w). This vector can be calculated using the following forward recurrence relation*

$$x_\tau^{LCP(w)} = \begin{cases} 0, & \tau \le 0; \\ \left(x_{\tau-1}^{LCP(w)}\right)_{x_\tau^{L,w}}^{x_\tau^{U,w}}, & \tau \ge 1. \end{cases} \tag{8}$$

Figure 1(b) illustrates the behavior of LCP(0). Note its similarity with Figure 1(a), but with the laziness in forward time instead of reverse time.

The computational demands of LCP(w) may initially seem prohibitive as τ grows, since calculating $x_\tau^{U,w}$ and $x_\tau^{L,w}$ requires solving convex optimizations of size $\tau + w$. However, it is possible to calculate $x_\tau^{U,w}$ and $x_\tau^{L,w}$ without using the full history. Lemma 10 in Appendix B implies that it is enough to use only the history since the most recent point when the solutions of (5) and (6) are either both increasing or both decreasing, if such a point exists. In practice, this period is typically less than a day due to diurnal traffic patterns, and so the convex optimization, and hence LCP(w), remains tractable even as τ grows. In Figure 1(b), each point is solved in under half a second.

Next, consider the cost incurred by LCP(w). Section VI discusses the cost in realistic settings, while in this section we focus on worst-case bounds, i.e., characterize the competitive ratio. The following theorem is proven in Appendix B.

Theorem 3. *$cost(X^{LCP(w)}) \le cost(X^*) + 2cost_s(X^*)$. Thus, LCP($w$) is 3-competitive for optimization (1). Further, for any finite w and $\epsilon > 0$ there exists an instance, with g_t of the form $x_t f(\lambda_t/x_t)$, such that LCP(w) attains a cost greater than $3 - \epsilon$ times the optimal cost.*

Note that Theorem 3 says that the competitive ratio is independent of the prediction window size w, the switching cost β, and is uniformly bounded above regardless of the form of the operating cost functions $g_t(\cdot)$. Surprisingly, this means that even the "myopic" LCP(0) is 3-competitive, regardless of $\{g_t(\cdot)\}$, despite having no information beyond the current timeslot. It is also surprising that the competitive ratio is tight regardless of w. Seemingly, for large w, LCP(w) should provide reduced costs. Indeed, for any particular workload, as w grows the cost decreases and eventually matches the optimal, when $w > T$. However, for any fixed w, there is a worst-case sequence of cost functions such that the competitive ratio is arbitrarily close to 3. Moreover, there is such a sequence of cost functions having the natural form $x_t f(\lambda_t/x_t)$, which corresponds to a time varying load λ being shared equally among x_t servers, which each has a time-invariant (though pathological) operating cost $f(\lambda)$ when serving load λ.

Finally, though 3-competitive may seem like a large gap, the fact that $cost(X^{LCP(w)}) \le cost(X^*) + 2cost_s(X^*)$ highlights that the gap will tend to be much smaller in practice, where the switching costs make up a small fraction of the total costs since dynamic right-sizing would tend to toggle servers once a day due to the diurnal traffic.

VI. Case Studies

In this section the goal is two-fold: The first is to evaluate the cost incurred by LCP(w) relative to the optimal solution and RHC(w) for realistic workloads. The second is more generally to illustrate the cost savings and energy savings that come from dynamic right-sizing in data centers. To accomplish these goals, we experiment using two real-world traces.

A. Experimental setup

Throughout the experimental setup, the aim is to choose parameters that provide conservative estimates of the cost savings from LCP(w) and right-sizing in general.

Cost benchmark: Current data centers typically do not use dynamic right-sizing and so to provide a benchmark against which LCP(w) is judged, we consider the cost incurred by a "static" right-sizing scheme for capacity provisioning. This chooses a constant number of servers that minimizes the costs incurred based on full knowledge of the entire workload. This policy is clearly not possible in practice, but it provides a conservative estimate of the savings from right-sizing since it uses perfect knowledge of all peaks and eliminates the need for overprovisioning in order to handle the possibility of flash crowds or other traffic bursts.

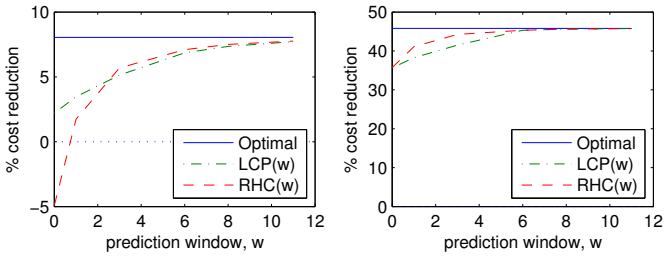
Cost function parameters: The cost is of the form (2), characterized by the parameters d_0, d_1, μ, p_t, e_0 and e_1, and the switching cost β. We normalize $\mu = 1$, $p_t = 1$ and choose units such that the fixed power is $e_0 = 1$. The load-dependent power is set to $e_1 = 0$, because the energy consumption of current servers is dominated by the fixed costs [2].

The delay cost d_1 reflects revenue lost due to customers being deterred by delay, or to violation of SLAs. We set $d_1/e_0 = 1$ for most experiments but consider a wide range of settings in Figure 8. The minimum perceptible delay is set to $d_0 = 1.5$ times the time to serve a single job. The value 1.5 is realistic or even conservative, since "valley filling" experiments similar to those of Section VI-B show that a smaller value would result in a significant added cost when using valley filling, which operators now do with minimal incremental cost.

The normalized switching cost β/e_0 measures the duration a server must be powered down to outweigh the switching cost. We use $\beta = 6e_0$, which corresponds to the energy consumption for one hour (six samples). This was chosen as an estimate of the time a server should sleep so that the wear-and-tear of power cycling matches that of operating [9].

Workload information: The workloads for these experiments are drawn from I/O traces of two real-world data centers. The first set of traces is from Hotmail, a large email service running on tens of thousands of servers. We used traces from 8 such servers over a 48-hour period, starting at midnight (PDT) on Monday August 4 2008 [5]. The second set of traces is taken from 6 RAID volumes at MSR Cambridge. The traced period was 1 week starting from 5PM GMT on the 22nd February 2007 [5]. These activity traces represent respectively a service used by millions of users and a small service used by hundreds of users. The traces are normalized such that the peak load is 100, which are shown in Figure 2. Notice that this normalization does not affect the experiment results since only the shape of trace matters for cost saving. Both sets of traces show strong diurnal properties and have peak-to-mean ratios (PMRs) of 1.64 and 4.64 for Hotmail and MSR respectively. Loads were averaged over disjoint 10 minute intervals.

The Hotmail trace contains significant nightly activity due to maintenance processes (backup, index creation etc) which is not shown fully in Figure 2(a). The data center, however, is provisioned for the peak foreground activity. This creates a dilemma: should our experiments include the maintenance activity or to remove it? Figure 5 illustrates the impact of this decision. If the spike is retained, it makes up nearly 12% of the total load and forces the static provisioning to use a much larger number of servers than if it were removed, making savings from dynamic right-sizing much more dramatic. To provide conservative estimates of the savings from right-sizing, we chose to trim the size of the spike to minimize the savings from right-sizing. This trimming makes the nightly background spike have value roughly equal to 100 (the peak foreground activity).



(a) Hotmail (b) MSR

Fig. 3. Impact of prediction window size on cost incurred by LCP(w).

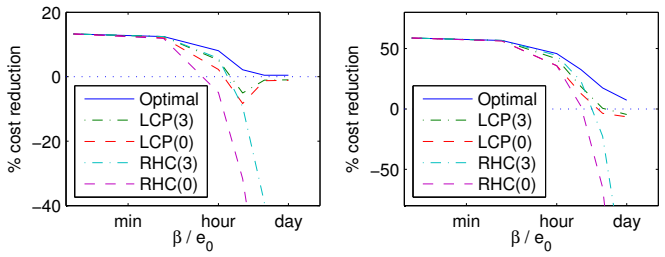
B. When is right-sizing beneficial?

Our experiments are organized in order to illustrate the impact of a wide variety of parameters on the cost-savings provided by dynamic right-sizing with LCP(w). The goal is to better understand when dynamic right-sizing can provide large enough cost-savings to warrant the extra implementation complexity. Remember that throughout, we have attempted to choose experimental settings so that the benefit of dynamic right-sizing is conservatively estimated.

The analytic results in Theorem 1 and Theorem 3 show that the performance of RHC(w) is more sensitive to the prediction window w and the switching cost β/e_0 while LCP(w) works well for general settings. The first two experiments consider the realistic cost saving from RHC(w) and LCP(w) with varying w and β/e_0 under real-world traces.

Impact of prediction window size: The first parameter we study is the impact of the predictability of the workload. In particular, depending on the workload, the prediction window w for which accurate estimates can be made could be on the order of tens of minutes or on the order of hours. Figure 3 illustrates its impact on the cost savings of RHC(w) and LCP(w), where the unit of w is one timeslot of is 10 minutes.

The first observation from Figure 3 is that the savings possible ("Optimal") in the MSR trace are larger than in the Hotmail trace. Second, the cost saving of LCP(w) is similar to RHC(w) when $w \geq 3$. However, RHC(w) can be much worse when $w \leq 2$, i.e., LCP(w) has more stable cost saving than RHC(w), which supports the analytic results very well. In both traces, a big fraction of the optimal cost savings is achieved by LCP(0), which uses only workload predictions about the current timeslot (10 minutes). The fact that this myopic algorithm provides significant gain over static provisioning is encouraging. Further, a prediction window that



(a) Hotmail (b) MSR

Fig. 4. Impact of switching cost, against time on a logarithmic scale.

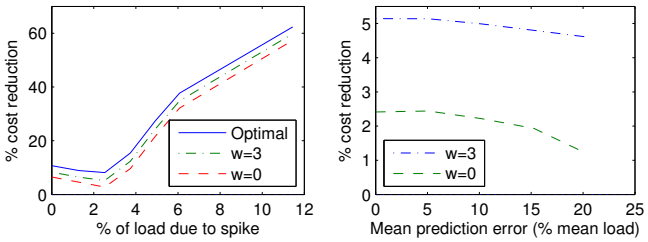
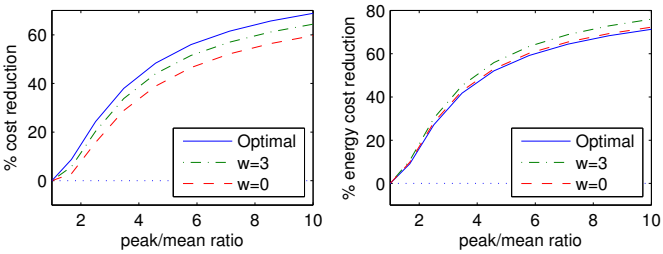


Fig. 5. Impact of overnight peak in the Hotmail workload.

Fig. 6. Impact of prediction error on LCP(w) under Hotmail workload.



(a) Total cost (b) Energy cost

Fig. 7. Impact of the peak-to-mean ratio of the workload on the total cost and energy cost incurred by LCP(w) in the Hotmail workload.

is approximately the size of $\beta = 6$ (i.e. one hour) gives nearly the optimal cost savings.

Impact of switching costs: One of the main worries when considering right-sizing is the switching cost of toggling servers, β, which includes the delay costs, energy costs, costs of wear-and-tear, and other risks involved. Thus, an important question to address is: "How large must switching costs be before the cost savings from right-sizing disappears?"

Figure 4 shows that significant gains are possible provided β is smaller than the duration of the valleys. Given that the energy costs, delay costs, and wear-and-tear costs are likely to be on the order of an hour, this implies that unless the risks associated with toggling a server are perceived to be extreme, the benefits from dynamic right-sizing are large in the MSR trace (high PMR case). Though the gains are smaller in the Hotmail case for large β, this is because the spike of background work splits an 8 hour valley into two short 4 hour valleys. If these tasks were shifted or balanced across the valley, the Hotmail trace would show significant cost reduction for much larger β, similarly to the MSR trace.

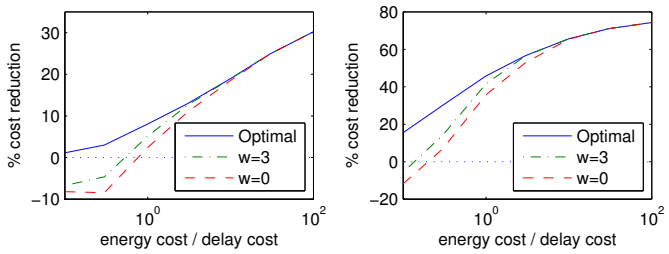
Another key observation is that the cost saving of LCP(w) is similar to that of RHC(w) when β is small or moderate. However, when β becomes big, LCP(w) is much better than RHC(w), which confirms that LCP(w) provides more stable cost saving than RHC(w). Since we use moderate prediction window and moderate switching cost as the default setting in our experiments, we will show only the LCP(w) results but not RHC(w) results for the remaining experiments.

Prediction error: The LCP(w) algorithm depends on having estimates for the arrival rate during the current timeslot as well as for w timeslots into the future. Our analysis in Section V assumes that these estimates are perfect, but of course in practice there are prediction errors. Figure 6 shows the cost reduction for the Hotmail trace when additive white Gaussian noise of increasing variance is added to the predictions used by LCP(w), including the workload in current timeslot. A variance of 0 corresponds to perfect knowledge of the near-future workload, and this figure shows that the performance does not degrade significantly when there is moderate uncertainty, which suggests that the assumptions of the analysis are not problematic. The plot for the MSR trace is qualitatively similar, although the actual cost savings are significantly larger.

Even very inexact knowledge of future workloads can be beneficial. Note that LCP(0)'s upper and lower bounds x^U and x^L implicitly assume that future workloads will be infinite and 0, respectively. Performance can be improved if either of these bounds can be tightened. For example, it may be possible to use daily trends to predict a lower bound on the load, while the upper bound is still taken to be infinite to allow for flash crowds. Given that prediction errors for real data sets tend to be small [8], [22], based on these plots, to simplify our experiments we allow LCP(w) perfect predictions.

Impact of peak-to-mean ratio (PMR): Dynamic right-sizing inherently exploits the gap between the peaks and valleys of the workload, and intuitively provides larger savings as that gap grows. To illustrate this, we artificially scaled the PMR of each trace and calculated the resulting savings. The results, in Figure 7, illustrate that the intuition holds for both cost savings and energy savings. The gain grows quickly from zero at PMR=1, to 5–10% at PMR$\approx$2 which is common in large data centers, to very large values for the higher PMRs common in small to medium sized data centers. This shows that, even for small data centers where the overhead of implementing right-sizing is amortized over fewer servers, there is a significant benefit in doing so. To provide some context for the monetary value of these savings, consider that a typical 50,000 server data center has an electricity bill of around $1 million/month [1].

The workload for the figure is generated from the Hotmail workload by scaling λ_t as $\hat{\lambda}_t = k(\lambda_t)^\alpha$, varying α and adjusting k to keep the mean constant. Note that though Figure 7 includes only the results for Hotmail, the resulting plot for the MSR trace is nearly identical. This highlights that the difference in cost savings observed between the two traces is primarily due to the fact that the PMR of the MSR trace is so much larger than that of the Hotmail trace.



(a) Hotmail (b) MSR

Fig. 8. Impact of increasing energy costs.

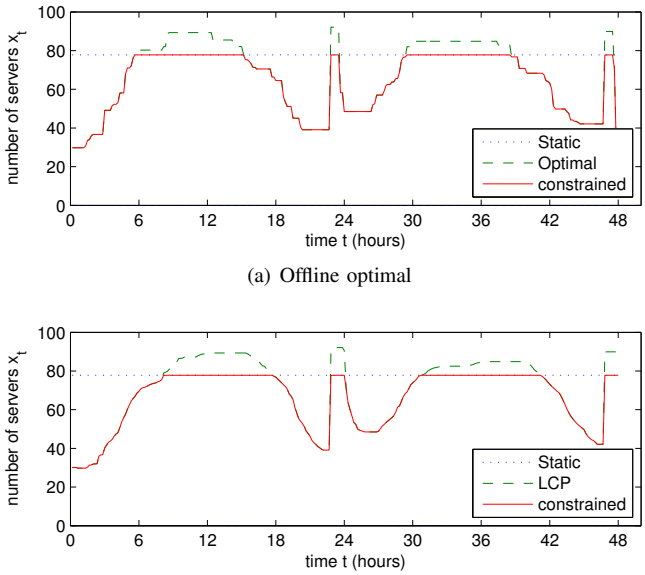
Impact of energy costs: Clearly the benefit of dynamic right-sizing is highly dependent on the cost of energy. As the economy is forced to move towards more expensive renewable energy sources, this cost will inevitably increase and Figure 8 shows how this increasing cost will affect the cost savings possible from dynamic right-sizing. Note that the cost savings from dynamic right-sizing grow quickly as energy costs rise. However, even when energy costs are quite small relative to delay costs, we see improvement in the case of the MSR workload due to its large PMR.

Impact of valley filling: A common alternative to dynamic right-sizing that is often suggested is to run very delay-insensitive maintenance/background processes during the periods of low load, known as "valley filling". Some applications have a huge amount of such background work, e.g., search engines tuning their ranking algorithms. If there is enough such background work, then the valleys can in principle be entirely filled and so the PMR≈ 1 and thus dynamic right-sizing is unnecessary. Thus, an important question is: "How much background work is enough to eliminate the cost savings from dynamic right-sizing?"

Figure 10 shows that, in fact, dynamic right-sizing provides cost savings even when background work makes up a significant fraction of the total load. For the Hotmail trace, significant savings are still possible when background load makes upwards of 10% of the total load, while for the MSR trace this threshold becomes nearly 60%. Note that Figure 10 results from considering "ideal" valley filling, which results in a perfectly flat load during the valleys, but does not give background processes lower queueing priority.

Impact of capacity limits: Energy-related expenses account for almost half of the cost of a data center. However, many of those, such as power supply and cooling infrastructure, depend on the peak power rather than the total energy. This raises the question of whether LCP increases the peak number of servers used, or equivalently the peak power consumption. The dotted line of Figure 9 shows the optimal static number of servers to use, which we call M, while the dashed line in Figure 9(a) shows the optimal number and that in Figure 9(b) shows the number used by LCP. Dynamic provisioning uses a higher peak number of servers, which would increase the cost.

However, the LCP algorithm (and off-line optimization) can impose an additional constraint $x_t \leq M_t = M$ to ensure that the infrastructure cost is not increased. Indeed, when M_t is independent of t the constrained optimum appears to have the



(a) Offline optimal

(b) LCP(0)

Fig. 9. Illustrations of (a) the offline optimal solution and (b) LCP(0) for the Hotmail workload described in Section VI with a sampling period of 10 minutes. The operating cost is as in Figure 1. The dotted line shows the optimal static number of servers to use, the dashed line shows the optimal or LCP provisioning without constraints, and the solid line shows the constrained optimum or constrained LCP solution. Notice that the constrained optimum turns out to correspond to simple clipping of the unconstrained optimum.

very simple form $\min(M, x_t^*)$ — the unconstrained optimum clipped to M — though this structure does not hold for general time-varying bounds. An example of x_t is shown by the solid lines in Figure 9. Despite the additional constraint, most of the potential gains are still realized by dynamic right sizing. In this case, the gain of the optimal solution drops from 8% to 6.5%, and that of LCP drops from 5.1% to 4.4%.

VII. Related work

Interest in right-sizing has been growing since [14] and [23] appeared early last decade. Approaches range from very "analytic" work focusing on developing algorithms with provable guarantees to "systems" work focusing on implementation. Early systems work such as [23] achieved substantial savings despite ignoring switching costs in their design. Other designs have focused on decentralized frameworks, e.g., [24], [25] and [26], as opposed to the centralized framework considered here. A recent survey is [27].

Related analytic work focusing on dynamic right-sizing includes [28], which reallocates resources between tasks within a data center, and [29], [30], which considers sleep of individual components, among others. Typically, approaches have applied optimization using queueing theoretic models, e.g., [31], [32], or control theoretic approaches, e.g., [33]–[35]. A recent survey of analytic work focusing on energy efficiency in general is [36]. Our work is differentiated from this literature by the generality of the model considered, which subsumes most common energy and delay cost models used by analytic researchers, and the fact that we provide worst-case guarantees for the cost of the algorithm, which is typically not possible for queueing or control theoretic based algorithms.

The model and algorithm introduced in this paper most closely ties to the online algorithms literature, specifically the classic rent-or-buy (or "ski rental") problem [37]. The best deterministic strategy for deciding when to turn off a single idle server (i.e., to stop "renting" and to "buy") is 2-competitive [38]. Additionally, there is a randomized algorithm which is asymptotically $e/(e-1)$-competitive [39]. These algorithms have been directly translated to the data-center context in [40], [41]. These papers assume that the power requirements of servers are fixed when they are on, and that there is no performance penalty for increasing the load on each server up to the maximum load it can support. In this setting, a judicious allocation of jobs to servers allows each server to perform an independent rent-or-buy decision.

The "lower envelope" algorithm of [29], [30], which generalizes the standard rent-or-buy algorithm, puts a device into deeper sleep mode m at a time t such that the optimal solution would use m if the idle period finished right after t. This is like LCP following x^U downward; in [29], [30] a device must be fully on to serve work, and so there is no equivalent to x^L forcing the system to turn on in stages.

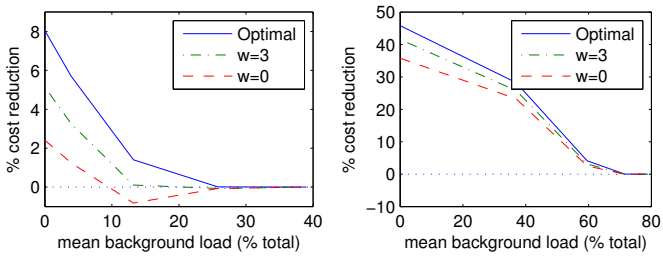
An important difference between these simple models and the current paper is that the cost of the "rent" action may change arbitrarily over time in the data center optimization problem. Problems with this sort of dynamics typically have competitive ratios greater than 2. For example, when rental prices vary in time, the competitive ratio is unbounded in general [42]. Further, for "metrical task systems" [43], which generalize rent-or-buy problems and the data center optimization problem, there are many algorithms available, but they typically have competitive ratios that are at best poly-logarithmic in the number of servers. Perhaps the most closely related prior work from this area is the page-placement problem (deciding on which server to store a file), which has competitive ratio 3 [44]. The page replacement-problem is nearly a discrete version of the data center optimization problem where the cost function is restricted to $f(x) = |x-1|$.

VIII. Extensions

The general model captures many applications other than the right-sizing problems in data centers. Intuitively, this model seeks to minimize the sum of a sequence of convex functions while "smooth" solutions are preferred. Other applications may be captured by this model, such as video streaming [45], in which encoding quality should vary depending on network bandwidth, but large changes in encoding quality are visually annoying to users.

A natural generalization of the framework of (1) is to make x_t a vector, corresponding to managing resources of multiple different types. This extension has many important applications, including joint capacity provision in geographically distributed data centers [25], automatically switched optical networks (ASONs) in which there is a cost for re-establishing a lightpath [46], and power generation with dynamic demand, since the cheapest types of generators typically have very high switching costs [47]

Unfortunately, LCP(w) does not naturally generalize to the case that x_t is a vector. Moreover, although the RHC(w) algorithm can be directly applied, it can perform poorly when x_t is a vector [21]. An improved algorithm for the case of heterogeneous resources with lookahead has been proposed in [21], but an algorithm without lookahead is still the subject of active research.



(a) Hotmail (b) MSR

Fig. 10. Impact of background processes. The improvement of LCP(w) over static provisioning as a function of the percentage of the workload that is background tasks.

IX. Summary and Concluding Remarks

This paper has provided a new online algorithm, LCP(w), for dynamic right-sizing in data centers. The algorithm is motivated by the structure of the optimal offline solution and guarantees cost no larger than 3 times the optimal cost, under very general settings — arbitrary workloads, and general delay cost and general energy cost models provided that they result in a convex operating cost. Further, in realistic settings the cost of LCP(w) is nearly optimal. Additionally, LCP(w) is simple to implement in practice and does not require significant computational overhead. Moreover, we contrast LCP(w) with the more traditional approach of receding horizon control RHC(w) and show that LCP(w) provides much more stable cost saving with general settings.

Additionally, the case studies used to evaluate LCP(w) highlight that the cost and energy savings achievable by dynamic right-sizing are significant. The case studies highlight that if a data center has PMR larger than 3, a cost of toggling a server of less than a few hours of server costs, and less than 40% background load then the cost savings from dynamic right-sizing can be conservatively estimated at larger than 15%. Thus, even if a data center is currently performing valley filling, it can still achieve significant cost savings by dynamic right-sizing.

Acknowledgements

This work was supported by NSF grants CCF 0830511, and CNS 0846025, Microsoft Research, the Lee Center for Advanced Networking, and ARC grant FT0991594.

References

[1] J. Hamilton, "Cost of power in large-scale data centers," http://perspectives.mvdirona.com/, Nov. 2009.
[2] L. A. Barroso and U. Hölzle, "The case for energy- proportional computing," *Computer*, vol. 40, no. 12, pp. 33–37, 2007.
[3] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proc. USENIX NSDI*, 2005, pp. 273–286.

[4] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, and F. Zhao, "Energy-aware server provisioning and load dispatching for connection-intensive internet services," in *Proc. USENIX NSDI*, 2008.
[5] E. Thereska, A. Donnelly, and D. Narayanan, "Sierra: a power-proportional, distributed storage system," Microsoft Research, Tech. Rep. MSR-TR-2009-153, 2009.
[6] H. Amur, J. Cipar, V. Gupta, G. R. Ganger, M. A. Kozuch, and K. Schwan, "Robust and flexible power-proportional storage," in *Proc. ACM SoCC*, 2010.
[7] 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.
[8] 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.
[9] P. Bodik, M. P. Armbrust, K. Canini, A. Fox, M. Jordan, and D. A. Patterson, "A case for adaptive datacenters to conserve energy and improve reliability," University of California at Berkeley, Tech. Rep. UCB/EECS-2008-127, 2008.
[10] "SPEC power data on SPEC website at http://www.spec.org."
[11] L. Kleinrock, *Queueing Systems Volume II: Computer Applications*. Wiley Interscience, 1976.
[12] R. Urgaonkar, U. Kozat, K. Igarashi, and M. Neely, "Dynamic resource allocation and power management in virtualized data centers," in *Proc. IEEE/IFIP NOMS*, Apr. 2010.
[13] R. Das, J. O. Kephart, C. Lefurgy, G. Tesauro, D. W. Levine, and H. Chan, "Autonomic multi-agent management of power and performance in data centers," in *Proc. Int. Joint Conf. Autonomous Agents and Multiagent Systems*, 2008.
[14] 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 SOSP*, 2001, pp. 103–116.
[15] L. L. H. Andrew, M. Lin, and A. Wierman, "Optimality, fairness and robustness in speed scaling designs," in *Proc. ACM SIGMETRICS*, 2010.
[16] L. Rao, X. Liu, L. Xie, and W. Liu, "Minimizing electricity cost: Optimization of distributed internet data centers in a multi-electricity-market environment," in *Proc IEEE INFOCOM*, Mar 2010.
[17] H. N. Van, F. Tran, and J.-M. Menaud, "Sla-aware virtual resource management for cloud infrastructures," in *IEEE Int. Conf. Comput. Inform. Technol. (CIT)*, OCT 2009.
[18] 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.
[19] 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.
[20] D. Q. Mayne and H. Michalska, "Receding horizon control of nonlinear systems," *IEEE Trans. Automatic Control*, vol. 35, no. 7, pp. 814–824, 1990.
[21] M. Lin, Z. Liu, A. Wierman, and L. L. H. Andrew, "Online algorithms for geographical load balancing," in *Proc. Int. Green Computing Conf.*, San Jose, CA, June 2012.
[22] 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*, Boston, MA, Sept. 2007.
[23] E. Pinheiro, R. Bianchini, E. Carrera, and T. Heath, "Load balancing and unbalancing for power and performacne in cluster-based systems," in *proc. Compilers and Operating Systems for Low Power*, 2001.
[24] B. Khargharia, S. Hariri, and M. Yousif, "Autonomic power and performance management for computing systems," *Cluster computing*, vol. 11, no. 2, pp. 167–181, Dec. 2007.
[25] Z. Liu, M. Lin, A. Wierman, S. H. Low, and L. L. H. Andrew, "Greening geographical load balancing," in *Proc. ACM SIGMETRICS*, 2011.
[26] A. Kansal, J. Liu, A. Singh, R. Nathuji, and T. Abdelzaher, "Semantic-less coordination of power management and application performance," in *ACM SIGOPS*, 2010, pp. 66–70.
[27] A. Beloglazov, R. Buyya, Y. C. Lee, and A. Zomaya, "A taxonomy and survey of energy-efficient data centers and cloud computing systems," *Advances in Computers*, vol. 82, pp. 47–111, 2011.
[28] C. G. Plaxton, Y. Sun, M. Tiwari, , and H. Vin, "Reconfigurable resource scheduling," in *ACM SPAA*, 2006.
[29] S. Irani, R. Gupta, and S. Shukla, "Competitive analysis of dynamic power management strategies for systems with multiple power savings states," in *Proc. Design, Automation, and Test in Europe*, 2002, p. 117.
[30] J. Augustine, S. Irani, and C. Swamy, "Optimal power-down strategies," *SIAM Journal on Computing*, vol. 37, no. 5, pp. 1499–1516, 2008.
[31] A. Gandhi, M. Harchol-Balter, R. Das, and C. Lefurgy, "Optimal power allocation in server farms," in *Proc. of ACM Sigmetrics*, 2009.
[32] A. Gandhi, V. Gupta, M. Harchol-Balter, and M. A. Kozuch, "Optimality analysis of energy-performance trade-off for server farm management," *Performance Evaluation*, vol. 67, no. 11, pp. 1155 – 1171, 2010.
[33] T. Horvath and K. Skadron, "Multi-mode energy management for multi-tier server clusters," in *Proc. PACT*, 2008.
[34] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam, "Managing server energy and operational costs in hosting centers," in *Proc. Sigmetrics*, 2005.
[35] R. Urgaonkar, U. C. Kozat, K. Igarashi, and M. J. Neely, "Dynamic resource allocation and power management in virtualized data centers," in *Proc. IEEE/IFIP NOMS*, Apr. 2010.
[36] S. Albers, "Energy-efficient algorithms," *Comm. of the ACM*, vol. 53, no. 5, pp. 86–96, 2010.
[37] A. Borodin and R. El-Yaniv, *Online computation and competitive analysis*. Cambridge University Press, 1998.
[38] A. R. Karlin, M. S. Manasse, L. Rudolph, and D. D. Sleator, "Competitive snoopy caching," *Algorithmica*, vol. 3, no. 1, pp. 77–119, 1988.
[39] A. R. Karlin, C. Kenyon, and D. Randall, "Dynamic TCP acknowledgement and other stories about $e/(e-1)$," in *Proc. ACM Symp. Theory of Computing (STOC)*, 2001.
[40] T. Lu and M. Chen, "Simple and effective dynamic provisioning for power-proportional data centers," in *Information Sciences and Systems (CISS), 2012 46th Annual Conference on*. IEEE, 2012, pp. 1–6.
[41] T. Lu, M. Chen, and L. L. H. Andrew, "Simple and effective dynamic provisioning for power-proportional data centers," *IEEE Trans. Parallel and Distributed Systems*, To appear. [Online]. Available: http://www.caia.swin.edu.au/cv/landrew/pubs/CSR.pdf
[42] M. Bienkowski, "Price fluctuations: To buy or to rent," in *Approximation and Online Algorithms*, 2008, pp. 25–36.
[43] A. Borodin, N. Linial, and M. E. Saks, "An optimal on-line algorithm for metrical task system," *J. ACM*, vol. 39, no. 4, pp. 745–763, 1992.
[44] D. L. Black and D. D. Sleator, "Competitive algorithms for replication and migration problems," Carnegie Mellon University, Tech. Rep. CMU-CS-89-201, 1989.
[45] M. Zink, O. Künzel, J. Schmitt, and R. Steinmetz, "Subjective impression of variations in layer encoded videos," in *Proc. Int. Conf. Quality of service*. Springer-Verlag, 2003, pp. 137–154.
[46] Y. Zhang, M. Murata, H. Takagi, and Y. Ji, "Traffic-based reconfiguration for logical topologies in large-scale WDM optical networks," *IEEE J. Lightwave Technology*, vol. 23, no. 10, p. 2854, 2005.
[47] S. Kaplan, "Power plants: Characteristics and costs," *Congressional Research Service*, 2008.

APPENDIX A
ANALYSIS OF THE OFFLINE OPTIMAL SOLUTION

In this section we will prove Lemma 1 and Theorem 2. Before beginning the proofs, let us first rephrase the data center optimization (1) as follows.

$$\begin{aligned} \text{minimize} \quad & \sum_{t=1}^{T} g_t(x_t) + \beta \sum_{t=1}^{T} y_t \\ \text{subject to} \quad & y_t \geq x_t - x_{t-1}, \quad x_t \geq 0, \quad y_t \geq 0. \end{aligned} \tag{9}$$

Next, we want to work with the dual of optimization (9). The Lagrangian (with respect to the first constraint) is

$$\begin{aligned} L(x,y,\nu) &= \sum_{t=1}^{T} g_t(x_t) + \beta \sum_{t=1}^{T} y_t + \sum_{t=1}^{T} \nu_t (x_t - x_{t-1} - y_t) \\ &= \sum_{t=1}^{T} (g_t(x_t) + (\beta - \nu_t) y_t) \\ &\quad + \sum_{t=1}^{T-1} (\nu_t - \nu_{t+1}) x_t + \nu_T x_T - \nu_1 x_0 \end{aligned}$$

and the dual function is $D(\nu) = \inf_{x \geq 0, y \geq 0} L(x,y,\nu)$.Since we are interested in the maximum of $D(\nu)$ in the dual problem,

we need only to consider the case $\beta - \nu_t \geq 0$. Then the dual function becomes

$$\begin{aligned} & D(\nu) \\ &= \inf_{x \geq 0} \left(\sum_{t=1}^{T} g_t(x_t) + \sum_{t=1}^{T-1} (\nu_t - \nu_{t+1}) x_t + \nu_T x_T - \nu_1 x_0 \right) \\ &= - \sum_{t=1}^{T-1} g_t^*(\nu_{t+1} - \nu_t) - g_T^*(-\nu_T) - \nu_1 x_0. \end{aligned}$$

where $g_t^*(\cdot)$ is the convex conjugate of function $g_t(\cdot)$ (enforcing $g_t(x_t) = \infty$ when $x_t < 0$). Therefore the corresponding dual problem of (9) is

$$\begin{aligned} \max & - \sum_{t=1}^{T-1} g_t^*(\nu_{t+1} - \nu_t) - g_T^*(-\nu_T) - \nu_1 x_0 \qquad (10) \\ \text{subject to } & 0 \leq \nu_t \leq \beta, \end{aligned}$$

where the complementary slackness conditions are

$$\begin{aligned} \nu_t(x_t - x_{t-1} - y_t) &= 0 \qquad (11) \\ (\beta - \nu_t) y_t &= 0, \qquad (12) \end{aligned}$$

and the feasibility condition is $x_t \geq 0$, $y_t \geq 0$, $y_t \geq x_t - x_{t-1}$ and $0 \leq \nu_t \leq \beta$.

Using the above, we now observe a relationship between the data center optimization in (9) and the upper and lower bounds, i.e., optimizations (6) and (5). Specifically, if $\nu_{\tau+1} = 0$ in a solution of optimization (10), then $\nu_1, \ldots, \nu_\tau$ is a solution to:

$$\begin{aligned} \max & - \sum_{t=1}^{\tau-1} g_t^*(\nu_{t+1} - \nu_t) - g_\tau^*(-\nu_\tau) - \nu_1 x_0 \qquad (13) \\ \text{subject to } & 0 \leq \nu_t \leq \beta, \end{aligned}$$

which is identical to the dual problem of optimization (5). Thus, the corresponding $x_1, \ldots, x_\tau$ is a solution to optimization (5). On the other hand, if $\nu_{\tau+1} = \beta$ in a solution of optimization (10), then $\nu_1, \ldots, \nu_\tau$ is a solution to:

$$\begin{aligned} \max & - \sum_{t=1}^{\tau-1} g_t^*(\nu_{t+1} - \nu_t) - g_\tau^*(\beta - \nu_\tau) - \nu_1 x_0 \qquad (14) \\ \text{subject to } & 0 \leq \nu_t \leq \beta. \end{aligned}$$

Let $\nu'_t = \beta - \nu_t$, which makes (14) become

$$\begin{aligned} \max & - \sum_{t=1}^{\tau-1} g_t^*(\nu'_t - \nu'_{t+1}) - g_\tau^*(\nu'_\tau) + \nu'_1 x_0 - \beta x_0 \\ \text{subject to } & 0 \leq \nu'_t \leq \beta. \qquad (15) \end{aligned}$$

It is easy to check that the corresponding $x_1, \ldots, x_\tau$ is a solution to optimization (6).

We require some notation and two technical lemmas before moving to the proofs of Lemma 1 and Theorem 2. Denote $h_{i,j}(x; x_S; x_E) = \sum_{t=i}^{j} g_t(x_t) + \beta(x_i - x_S)^+ + \beta \sum_{t=i+1}^{j} (x_t - x_{t-1})^+ + \beta(x_E - x_j)^+$ for $j \geq i$, and $h'_{i,j}(x; x_S; x_E) = \sum_{t=i}^{j} g_t(x_t) + \beta(x_S - x_i)^+ + \beta \sum_{t=i+1}^{j} (x_{t-1} - x_t)^+ + \beta(x_j - x_E)^+$ for $j \geq i$. Then the objective of (5) is $h_{1,\tau}(x; x_0; 0)$. Similarly, the objective of (6) is $h'_{1,\tau}(x; x_0; x_M)$ where

$$x_M = \max(\max_{\tau,t} x^U_{\tau,t}, \max_t x^*_t). \qquad (16)$$

The first lemma is to connect $h_{i,j}$ and $h'_{i,j}$:

Lemma 3. *Given x_S and x_E, any solution minimizing $h_{i,j}(x; x_S; x_E)$ also minimizes $h'_{i,j}(x; x_S; x_E)$ and vice versa.*

Proof:

$$\begin{aligned} & h_{i,j}(x; x_S; x_E) - h'_{i,j}(x; x_S; x_E) \\ =& \beta(x_i - x_S) + \beta \sum_{t=i+1}^{j} (x_t - x_{t-1}) + \beta(x_E - x_j) \\ =& \beta(x_E - x_S), \end{aligned}$$

which is a constant. Thus any minimizer of $h_{i,j}(x; x_S; x_E)$ or $h'_{i,j}(x; x_S; x_E)$ is also a minimizer of the other. ∎

Lemma 4. *Given x_S and $x_E \leq \hat{x}_E$, let $X = (x_i, \ldots, x_j)$ minimize $h_{i,j}(x; x_S; x_E)$, then there exists an $\hat{X} = (\hat{x}_i, \ldots, \hat{x}_j)$ minimizing $h_{i,j}(x; x_S; \hat{x}_E)$ such that $X \leq \hat{X}$.*

Proof: If there is more than one solution minimizing $h_{i,j}(x; x_S; \hat{x}_E)$, let $\hat{X}$ be a solution with greatest $\hat{x}_j$. We will first argue that $x_j \leq \hat{x}_j$. By definition, we have $h_{i,j}(X; x_S; x_E) \leq h_{i,j}(\hat{X}; x_S; x_E)$ and $h_{i,j}(\hat{X}; x_S; \hat{x}_E) \leq h_{i,j}(X; x_S; \hat{x}_E)$. Note that if the second inequality is an equality, then $x_j \leq \hat{x}_j$. Otherwise, sum the two inequalities, to obtain the strict inequality

$$(x_E - x_j)^+ + (\hat{x}_E - \hat{x}_j)^+ < (\hat{x}_E - x_j)^+ + (x_E - \hat{x}_j)^+.$$

Since $x_E \leq \hat{x}_E$, we can conclude that $x_j < \hat{x}_j$. Therefore, we always have $x_j \leq \hat{x}_j$.

Next, recursively consider the subproblem $h_{i,j-1}(\cdot)$ with $x_E = x_j$ and $\hat{x}_E = \hat{x}_j$. The same argument as above yields that $x_{j-1} \leq \hat{x}_{j-1}$. This continues and we can conclude that $(x_i, \ldots, x_j) \leq (\hat{x}_i, \ldots, \hat{x}_j)$. ∎

Lemma 4 shows that the optimizations in our paper have a unique maximal solution, as follows.

Corollary 1. *If there is more than one solution to optimization problem (1), (4), (5) or (6), there exists a maximum solution which is not less than others elementwise.*

Proof: Let us consider problem (1) with multiple solutions and prove the claim by induction. Assume that x^* is a solution with the first τ ($\tau \geq 1$) entries not less than those in other solutions, but $x^*_{\tau+1} < \hat{x}^*_{\tau+1}$ where $\hat{x}^*$ is a solution with the greatest $(\tau+1)$th entry. Since $(x^*_1, \ldots, x^*_\tau)$ minimizes $h_{1,\tau}(x; x_0; x^*_{\tau+1})$, Lemma 4 implies there exists a solution $(\bar{x}^*_1, \ldots, \bar{x}^*_\tau)$ minimizing $h_{1,\tau}(x; x_0; \hat{x}^*_{\tau+1})$ which is not less than $(x^*_1, \ldots, x^*_\tau)$. Thus we can replace $(\hat{x}^*_1, \ldots, \hat{x}^*_\tau)$ in $\hat{x}^*$ by $(\bar{x}^*_1, \ldots, \bar{x}^*_\tau)$ to get a solution with the first $\tau + 1$ entries not less than other solutions. The proof for optimization problem (4), (5) and (6) are similar and thus omitted. ∎

We now complete the proofs of Lemma 1 and Theorem 2.

Proof of Lemma 1: Let $X^L_\tau = (x^L_{\tau,1}, x^L_{\tau,2}, \ldots, x^L_{\tau,\tau})$ be the solution of optimization (5) at time τ and define X^U_τ symmetrically for (6). Also, let $X^*_\tau = (x^*_1, \ldots, x^*_\tau)$ be the

first τ entries of the offline solution to optimization (1).

We know that X^L_τ minimizes $h_{1,\tau}(x; x_0; 0)$ and X^*_τ minimizes $h_{1,\tau}(x; x_0; x^*_{\tau+1})$. Since $x^*_{\tau+1} \geq 0$, Lemma 4 implies $X^*_\tau \geq X^L_\tau$, and thus, in particular, the last entry satisfies $x^*_\tau \geq x^L_{\tau,\tau}$.

Symmetrically, X^U_τ minimizes $h'_{1,\tau}(x; x_0; x_M)$ where x_M satisfies (16). By Lemma 3, X^U_τ also minimizes $h_{1,\tau}(x; x_0; x_M)$. Since $x^*_{\tau+1} \leq x_M$, Lemma 4 implies that $X^*_\tau \leq X^U_\tau$, and thus $x^*_\tau \leq x^U_{\tau,\tau}$. ∎

Proof of Theorem 2: As a result of Lemma 1, we know that $x^*_\tau \in [x^L_\tau, x^U_\tau]$ for all τ. Further, if $x^*_\tau > x^*_{\tau+1}$, by the complementary slackness condition (11), we have that $\nu_{\tau+1} = 0$. Thus, in this case, x^*_τ solves optimization (5) for the lower bound, i.e., $x^*_\tau = x^L_\tau$. Symmetrically, if $x^*_\tau < x^*_{\tau+1}$, we have that complementary slackness condition (12) gives $\nu_{\tau+1} = \beta$ and so x^*_τ solves optimization (6) for the upper bound, i.e., $x^*_\tau = x^U_\tau$. Thus, whenever x^*_τ is increasing/decreasing it must match the upper/lower bound, respectively. ∎

APPENDIX B
ANALYSIS OF LAZY CAPACITY PROVISIONING, LCP(w)

In this section we will prove Lemma 2 and Theorem 3.

Proof of Lemma 2: First, we prove that $x^{L,w}_\tau \leq x^*_\tau$. By definition, $x^{L,w}_\tau = x^L_{\tau+w,\tau}$, and so it belongs to a solution minimizing $h_{1,\tau+w}(x; x_0; 0)$. Further, we can view the optimal x^*_τ as an entry in a solution minimizing $h_{1,\tau+w}(x; x_0; x^*_{\tau+w+1})$. From these two representations, we can apply Lemma 4, to conclude that $x^{L,w}_\tau \leq x^*_\tau$.

Next, we prove that $x^L_\tau \leq x^{L,w}_\tau$. To see this we notice that x^L_τ is the last entry in a solution minimizing $h_{1,\tau}(x; x_0; 0)$. And we can view $x^{L,w}_\tau$ as an entry in a solution minimizing $h_{1,\tau}(x; x_0; x^L_{\tau+w,\tau+1})$ where $x^L_{\tau+w,\tau+1} \geq 0$. Based on Lemma 4 we get $x^L_\tau \leq x^{L,w}_\tau$.

The proof that $x^*_\tau \leq x^{U,w}_\tau \leq x^U_\tau$ is symmetric. ∎

From the above lemma, we immediately obtain an extension of the characterization of the offline optimum.

Corollary 2. *The optimal solution of the data center optimization (1) satisfies the following backwards recurrence relation*

$$x^*_\tau = \begin{cases} 0, & \tau > T; \\ (x^*_{\tau+1})^{x^{U,w}_\tau}_{x^{L,w}_\tau}, & \tau \leq T. \end{cases} \tag{17}$$

Moving to the proof of Theorem 3, the first step is to use the above lemmas to characterize the relationship between $x^{LCP(w)}_\tau$ and x^*_τ. Note that $x^{LCP(w)}_0 = x^*_0 = x^{LCP(w)}_{T+1} = x^*_{T+1} = 0$.

Lemma 5. *Consider the timeslots $0 = t_0 < t_1 < \ldots < t_m = T$ such that $x^{LCP(w)}_{t_i} = x^*_{t_i}$. Then, during each segment (t_{i-1}, t_i), either*

(i) $x^{LCP(w)}_t > x^*_t$ and both $x^{LCP(w)}_t$ and x^*_t are non-increasing for all $t \in (t_{i-1}, t_i)$, or

(ii) $x^{LCP(w)}_t < x^*_t$ and both $x^{LCP(w)}_t$ and x^*_t are non-decreasing for all $t \in (t_{i-1}, t_i)$.

Proof: The result follows from the characterization of the offline optimal solution in Corollary 2 and the definition of LCP(w). Given that both the offline optimal solution and LCP(w) are non-constant only for timeslots when they are equal to either $x^{U,w}_t$ or $x^{L,w}_t$, we know that at any time t_i where $x^{LCP(w)}_{t_i} = x^*_{t_i}$ and $x^{LCP(w)}_{t_i+1} \neq x^*_{t_i+1}$, we must have that both $x^{LCP(w)}_{t_i}$ and $x^*_{t_i}$ are equal to either $x^{U,w}_{t_i}$ or $x^{L,w}_{t_i}$.

Now we must consider two cases. First, consider the case that $x^{LCP(w)}_{t_i+1} > x^*_{t_i+1}$. It is easy to see that $x^{LCP(w)}_{t_i+1}$ doesn't match the lower bound since $x^*_{t_i+1}$ is not less than the lower bound. Thus $x^{LCP(w)}_{t_i} \geq x^{LCP(w)}_{t_i+1}$ since, by definition, LCP(w) will never choose to increase the number of servers it uses unless it matches the lower bound. Consequently, it must be that $x^*_{t_i} = x^{LCP(w)}_{t_i} \geq x^{LCP(w)}_{t_i+1} > x^*_{t_i+1}$. Since x^* is decreasing, both $x^{LCP(w)}_{t_i}$ and $x^*_{t_i}$ match the lower bound. Further, the next time, t_{i+1}, when the optimal solution and LCP(w) match is the next time either the number of servers in LCP(w) matches the lower bound $x^{L,w}_t$ or the next time the number of servers in the optimal solution matches the upper bound $x^{U,w}_t$. Thus, until that point, LCP(w) cannot increase the number of servers (since this happens only when it matches the lower bound) and the optimal solution cannot increase the number of servers (since this happens only when it matches the upper bound). This completes the proof of part (i) of the Lemma. The proof of part (ii) is symmetric. ∎

Given Lemma 5, we bound the switching cost of LCP(w).

Lemma 6. $cost_s(X^{LCP(w)}) = cost_s(X^*)$.

Proof: Consider the sequence of times $0 = t_0 < t_1 < \ldots < t_m = T$ such that $x^{LCP(w)}_{t_i} = x^*_{t_i}$ identified in Lemma 5. Then, each segment (t_{i-1}, t_i) starts and ends with the same number of servers being used under both LCP(w) and the optimal solution. Additionally, the number of servers is monotone for both LCP(w) and the optimal solution, thus the switching cost incurred by LCP(w) and the optimal solution during each segment is the same. ∎

Next, we bound the operating cost of LCP(w).

Lemma 7. $cost_o(X^{LCP(w)}) \leq cost_o(X^*) + \beta \sum_{t=1}^{T} |x^*_t - x^*_{t-1}|$.

Proof: Consider the sequence of times $0 = t_0 < t_1 < \ldots < t_m = T$ such that $x^{LCP(w)}_{t_i} = x^*_{t_i}$ identified in Lemma 5, and consider specifically one of these intervals (t_{i-1}, t_i) such that $x^{LCP(w)}_{t_{i-1}} = x^*_{t_{i-1}}$, $x^{LCP(w)}_{t_i} = x^*_{t_i}$.

There are two cases in the proof: (i) $x^{LCP(w)}_t > x^*_\tau$ for all $\tau \in (t_{i-1}, t_i)$ and (ii) $x^{LCP(w)}_t < x^*_\tau$ for all $t \in (t_{i-1}, t_i)$.

We handle *case (i)* first. Our goal is to prove that

$$\sum_{t=t_{i-1}+1}^{t_i} g_t(x^{LCP(w)}_t) \leq \sum_{t=t_{i-1}+1}^{t_i} g_t(x^*_t) + \beta |x^*_{t_{i-1}} - x^*_{t_i}|. \tag{18}$$

Define $X_\tau = (x_{\tau,1}, \ldots, x_{\tau,\tau-1}, x^{LCP(w)}_\tau)$ where $(x_{\tau,1}, \ldots, x_{\tau,\tau-1})$ minimizes $h'_{1,\tau-1}(x; x_0; x^{LCP(w)}_\tau)$.

Additionally, define $X'_\tau = (x'_{\tau,1}, \ldots, x'_{\tau,\tau})$ as the solution minimizing $h'_{1,\tau}(x; x_0; x^{LCP(w)}_\tau)$; by Lemma 3 this also minimizes $h_{1,\tau}(x; x_0; x^{LCP(w)}_\tau)$.

We first argue that $x'_{\tau,\tau} = x^{LCP(w)}_\tau$ via a proof by contradiction. Note that if $x'_{\tau,\tau} > x^{LCP(w)}_\tau$, based on similar argument in the proof of Theorem 2, we have $x'_{\tau,\tau} = x^L_\tau$, which contradicts the fact that $x'_{\tau,\tau} > x^{LCP(w)}_\tau \geq x^L_\tau$. Second,

if $x'_{\tau,\tau} < x_\tau^{LCP(w)}$, then we can follow a symmetric argument to arrive at a contradiction. Thus $x'_{\tau,\tau} = x_\tau^{LCP(w)}$.

Consequently, $x'_{\tau,t} = x_{\tau,t}$ for all $t \in [1,\tau]$ and we get

$$h'_{1,\tau}(X'_\tau; x_0; x_M) = h'_{1,\tau}(X_\tau; x_0; x_M) \tag{19}$$

Next, let us consider $X_{\tau+1} = (x_{\tau+1,1}, \ldots, x_{\tau+1,\tau}, x_{\tau+1}^{LCP(w)})$ where $(x_{\tau+1,1}, \ldots, x_{\tau+1,\tau})$ minimizes $h'_{1,\tau}(x; x_0; x_{\tau+1}^{LCP(w)})$. Recalling $x_t^{LCP(w)}$ is non-increasing in case (i) by Lemma 5, we have $X_{\tau+1} \le X'_\tau$ by Lemma 4. In particular, $x_{\tau+1,\tau} \le x_\tau^{LCP(w)}$. Thus

$$\begin{aligned} &h'_{1,\tau+1}(X_{\tau+1}; x_0; x_M) \\ \ge &h'_{1,\tau}((x_{\tau+1,1}, \ldots, x_{\tau+1,\tau}); x_0; x_M) + g_{\tau+1}(x_{\tau+1}^{LCP(w)}) \\ = &h'_{1,\tau}((x_{\tau+1,1}, \ldots, x_{\tau+1,\tau}); x_0; x_\tau^{LCP(w)}) + g_{\tau+1}(x_{\tau+1}^{LCP(w)}) \end{aligned} \tag{20}$$

By definition of X'_τ, we get

$$\begin{aligned} &h'_{1,\tau}((x_{\tau+1,1}, \ldots, x_{\tau+1,\tau}); x_0; x_\tau^{LCP(w)}) \\ \ge &h'_{1,\tau}(X'_\tau; x_0; x_\tau^{LCP(w)}) \ge h'_{1,\tau}(X'_\tau; x_0; x_M) \end{aligned} \tag{21}$$

Combining equations (19), (20) and (21), we obtain

$$h'_{1,\tau+1}(X_{\tau+1}; x_0; x_M) \ge h'_{1,\tau}(X_\tau; x_0; x_M) + g_{\tau+1}(x_{\tau+1}^{LCP(w)}).$$

By summing this equality for $\tau \in [t_{i-1}, t_i)$, we have

$$\begin{aligned} &\sum_{t=t_{i-1}+1}^{t_i} g_t(x_t^{LCP(w)}) \\ \le &h'_{1,t_i}(X_{t_i}; x_0; x_M) - h'_{1,t_{i-1}}(X_{t_{i-1}}; x_0; x_M). \end{aligned}$$

Since $x_{t_{i-1}}^{LCP(w)} = x^*_{t_{i-1}}$, $x_{t_i}^{LCP(w)} = x^*_{t_i}$, we know that both $X_{t_{i-1}}$ and X_{t_i} are prefixes[2] of the offline solution x^*, thus $X_{t_{i-1}}$ is the prefix of X_{t_i}. Expanding out $h'(\cdot)$ in the above inequality gives (18), which completes case (i).

In *case (ii)*, i.e., segments where $x_t^{LCP(w)} < x^*_\tau$ for all $t \in (t_{i-1}, t_i)$, a parallel argument shows that (18) again holds.

To complete the proof we combine the results from case (i) and case (ii), summing equation (18) over all segments (and the additional times when $x_t^{LCP(w)} = x^*_t$). ∎

We can now prove the competitive ratio in Theorem 3.

Lemma 8. *$cost(X^{LCP(w)}) \le cost(X^*) + 2cost_s(X^*)$. Thus, LCP(w) is 3-competitive for the data center optimization (1).*

Proof: Combining Lemma 7 and Lemma 6 gives that $cost(X^{LPC(w)}) \le cost(X^*) + \beta|x^*_t - x^*_{t-1}|$. Note that, because both LCP(w) and the optimal solution start and end with zero servers on, we have $\sum_{t=1}^T |x^*_t - x^*_{t-1}| = 2\sum_{t=1}^T (x^*_t - x^*_{t-1})^+$, which completes the proof. ∎

All that remains for the proof of Theorem 3 is to prove that the bound of 3 on the competitive ratio is tight.

Lemma 9. *The competitive ratio of LCP(w) is at least 3.*

Proof: The following is a family of instances parameterized by n and $m \ge 2$, whose competitive ratios approach the bound of 3. The operating cost for each server is defined as $f(z) = z^m + f_0$ for $0 \le z \le 1$ and $f(z) = \infty$ otherwise, whence $g_t(x_t) = x_t f(\lambda_t/x_t)$. The switching cost is $\beta = 0.5$. Let $\delta \in (1, 1.5)$ be such that $n = \log_\delta \frac{1}{\delta - 1}$. The arrival rate at time i is $\lambda_i = \delta^{i-1}$ for $1 \le i \le n$, and $\lambda_i = 0$ for $n < i \le T$, where $T > \beta/f_0 + n$ with $f_0 = \frac{\beta(\delta^m - 1)}{n(\delta^{mn} - 1)}$.

For the offline optimization, denote the solution by vector x^*. First note that $x^*_i = x^*_n$ for $i \in [1, n]$ since x^*_i is non-decreasing for $i \in [1, n]$ and, for the above f,

$$\frac{d}{dx}[xf(\lambda_i/x)] < 0 \text{ for } x \in [\lambda_i, x^*_n]. \tag{22}$$

Moreover, $x^*_i = 0$ for $i \in [n+1, T]$. Hence the minimum cost is $\sum_{i=1}^n \frac{\lambda_i^m}{(x^*_n)^{m-1}} + (nf_0 + \beta)x^*_n$. Then by the first order (stationarity) condition we get

$$(x^*_n)^{-m} = \frac{(nf_0 + \beta)(\delta^m - 1)}{(m-1)(\delta^{mn} - 1)}, \tag{23}$$

which is smaller than λ_n^{-m} because $nf_0 + \beta \le 1$, $m - 1 \ge 1$ and $\delta > 1$. Thus it is feasible ($f(\lambda_i/x^*_n)$ is finite). The cost for the offline optimal solution is then

$$C^* = \frac{m}{m-1}(nf_0 + \beta)x^*_n.$$

We consider $LCP(w)$ with $w = 0$ before considering the general case. Let $C_{[i,j]}$ denote the cost of LCP(0) on $[i, j]$.

For the online algorithm LCP(0), denote the result by vector $\hat{x}$. We know $\hat{x}_i$ is actually matching x_i^L in $[1, n]$ (x_i^U is not less than $x^*_i = x^*_n$), thus $\hat{x}_i$ is non-decreasing for $i \in [1, n]$ and $\hat{x}_n = x^*_n$. By the same argument as for x^*_n, we have

$$(\hat{x}_\tau)^{-m} = \frac{(\tau f_0 + \beta)(\delta^m - 1)}{(m-1)(\delta^{m\tau} - 1)} \tag{24}$$

Thus the cost for LCP(0) in $[1, n]$ is

$$\begin{aligned} C_{[1,n]} &= \sum_{\tau=1}^n \left(\frac{\lambda_\tau^m}{(\hat{x}_\tau)^{m-1}} + f_0 \hat{x}_\tau \right) + \beta x^*_n \\ &> \sum_{\tau=1}^n \delta^{m(\tau-1)} \left(\frac{(\tau f_0 + \beta)(\delta^m - 1)}{(m-1)(\delta^{m\tau} - 1)} \right)^{\frac{m-1}{m}} + \beta x^*_n \\ &> \left(\frac{\beta(\delta^m - 1)}{m-1} \right)^{\frac{m-1}{m}} \sum_{\tau=1}^n \delta^{\tau - m} + \beta x^*_n \\ &= \left(\frac{\beta(\delta^m - 1)}{m-1} \right)^{\frac{m-1}{m}} \frac{\delta^n - 1}{(\delta - 1)\delta^{m-1}} + \beta x^*_n \end{aligned}$$

Thus by (23)

$$\begin{aligned} \frac{C_{[1,n]}}{x^*_n} &> \frac{\delta^n - 1}{(\delta^{mn} - 1)^{1/m}} \cdot \frac{\beta(\delta^m - 1)}{(m-1)(\delta - 1)\delta^{m-1}} + \beta \\ &> \frac{\delta^n - 1}{\delta^n} \cdot \frac{\beta(\delta^m - 1)}{(m-1)(\delta - 1)\delta^{m-1}} + \beta \end{aligned}$$

As $n \to \infty$, both $\delta \to 1$ and $(\delta^n - 1)/\delta^n \to 1$, since $n = \log_\delta \frac{1}{\delta - 1}$. Since m is independent of δ, L'Hospital's Law gives

$$\begin{aligned} \lim_{n\to\infty} \frac{C_{[1,n]}}{x^*_n} &\ge \lim_{\delta \to 1} \frac{\beta m \delta^{m-1}}{(m-1)(m\delta^{m-1} - (m-1)\delta^{m-2})} + \beta \\ &= \frac{m}{m-1}\beta + \beta \end{aligned}$$

Now let us calculate the cost for LCP(0) in $[n+1, T]$.

For $\tau > n$, by LCP(0), we know that x_τ will stay constant

[2]That is, $X_{t_{i-1}}$ is the first t_{i-1} components of x^* and X_{t_i} is the first t_i components of x^*.

until it hits the upper bound. Let $X_\tau = \{x_{\tau,t}\}$ be the solution of optimization (6) in $[1,\tau]$ for any $\tau > n$. We now prove that for τ such that $(\tau-n)f_0 < \beta$, we have $x_{\tau,\tau} \geq x_n^*$, and thus $\hat{x}_\tau = x_n^*$.

Note that $x_{\tau,n} \geq x_n^*$ since x_n^* belongs to the lower bound. Given $x_{\tau,n}$, we know that $x_{\tau,n+1},\ldots,x_{\tau,\tau}$ is the solution to the following problem:

$$\begin{array}{ll}\text{minimize} & \displaystyle\sum_{t=n+1}^{\tau} f_0 X_{\tau,t} + \beta \sum_{t=n+1}^{\tau} (X_{\tau,t-1} - X_{\tau,t})^+ \\ \text{subject to} & X_{\tau,t} \geq 0\end{array}$$

Let $x_{\min} = \min\{x_{\tau,n},\ldots,x_{\tau,\tau}\}$. Then

$$\begin{aligned} & \sum_{t=n+1}^{\tau} f_0 x_{\tau,t} + \beta \sum_{t=n+1}^{\tau} (x_{\tau,t-1} - x_{\tau,t})^+ \\ \geq & \sum_{t=n+1}^{\tau} f_0 x_{\min} + \beta(x_{\tau,n} - x_{\min}) \\ \geq & (\tau-n) f_0 x_{\tau,n}\end{aligned}$$

The last inequality is obtained by substituting $\beta > (\tau-n)f_0$. We can see that $x_{\tau,i} = x_{\tau,n} (i \in [n+1,\tau])$ is a solution, thus $x_{\tau,\tau} \geq x_n^*$. (Recall that, if there are multiple solutions, we take the maximum one). Therefore, we have

$$C_{[n+1,\tau]} = (\tau-n) f_0 x_n^*.$$

Since $f_0 \to 0$ as $\delta \to 1$, we can find an τ so that $(\tau-n)f_0 \to \beta$, and thus

$$C_{[n+1,\tau]} \to \beta x_n^*. \tag{25}$$

By combining the cost for LCP(0) in $[1,n]$ and $[n+1,T]$, we have

$$C_{[1,T]}/C^* \geq \frac{\frac{m}{m-1}\beta + 2\beta}{\frac{m}{m-1}(nf_0+\beta)} = \frac{3-2/m}{1+nf_0/\beta}.$$

Using the relationship between n, f_0 and δ, we can choose a large enough m and n to make this arbitrarily close to 3. This finishes the proof for LCP(0).

Now let us consider LCP(w) for $w > 0$. Denote the solution of LCP(w) by $\hat{x}'$. At time $\tau \in [1, n-w]$, LCP(w) solves the same optimization problem as LCP(0) does at $\tau + w$. Thus $\hat{x}'_\tau = \hat{x}_{\tau+w}$ of LCP(0). Thus

$$\begin{aligned} C'_{[1,n-w]} &= \sum_{\tau=1}^{n-w} \left(\frac{\lambda_\tau^m}{(\hat{x}'_\tau)^{m-1}} + f_0 \hat{x}'_\tau \right) + \beta x_n^* \\ &= \sum_{\tau=1+w}^{n} \left(\frac{1}{\delta^{wm}} \frac{\lambda_\tau^m}{(\hat{x}_\tau)^{m-1}} + f_0 \hat{x}_\tau \right) + \beta x_n^* \\ &> \frac{1}{\delta^{wm}} C_{[1+w,n]}\end{aligned}$$

By pushing $n \to \infty$, and hence $\delta \to 1$, we have $C'_{[1,n-w]}/C_{[1,n]} \to 1$.

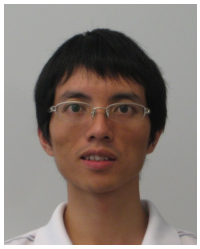
And for $\tau > n+1$, if $f_0(\tau-n) < \beta$, then $C'_{[n+1,\tau-w]} = (\tau-w-n)f_0 x_n^*$. By pushing $\delta \to 1$, we can find a τ such that $C'_{[n+1,\tau-w]} \to \beta x_n^*$, the same as (25). Therefore, as $\delta \to 1$, we have $C'_{[1,T]}/C_{[1,T]} \to 1$, thus the supremum over arbitrarily large $m \geq 2$ and arbitrarily small $\delta > 1$ of the competitive ratio of LCP(w) on this family is also 3. ∎

Finally, the following lemma ensures that the optimizations solved by LCP(w) at each timeslot τ remain small.

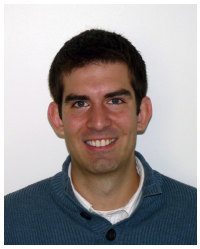
Lemma 10. *If there exists an index $\tau \in [1,t-1]$ such that $x^U_{t,\tau+1} < x^U_{t,\tau}$ or $x^L_{t,\tau+1} > x^L_{t,\tau}$, then $(x^U_{t,1},\ldots,x^U_{t,\tau}) = (x^L_{t,1},\ldots,x^L_{t,\tau})$. No matter what the future functions $g_i(\cdot)$ are, solving either (5) or (6) in $[1,t']$ for $t' > t$ is equivalent to solving two optimizations: one over $[1,\tau]$ with initial condition x_0 and final condition $x^U_{t,\tau}$ and the second over $[\tau+1,t']$ with initial condition $x^U_{t,\tau}$.*

Proof: Consider the case $x^L_{t,\tau+1} > x^L_{t,\tau}$. By the complementary slackness conditions (12), the corresponding dual variable $\nu_{\tau+1} = \beta$. Based on the argument in the proof of Theorem 2, we know the dual problem up to time τ is (14), which is identical to (15), the dual problem of (6). Since the optimal x_i depends only on ν_i and ν_{i+1}, the optimal x_i for $i \leq \tau$ are completely determined by ν_i for $i \in [1,\tau+1]$, with $\nu_{\tau+1} = \beta$. Hence $(x^L_{t,1},\ldots,x^L_{t,\tau}) = (x^U_{t,1},\ldots,x^U_{t,\tau})$. The proof for the case $x^U_{t,\tau+1} < x^U_{t,\tau}$ is symmetric.

Notice that $(x^U_{t,1},\ldots,x^U_{t,t})$ minimizes $h'_{1,t}(x;x_0;x_M)$ and thus $h_{1,t}(x;x_0;x_M)$ based on Lemma 3, $(x^L_{t,1},\ldots,x^L_{t,t})$ minimizes $h_{1,t}(x;x_0;0)$. No matter what the future functions $g_i(\cdot)$ are, the first t entries of its solution must minimize $h_{1,t}(x;x_0;x_{t+1})$ for some $x_{t+1} \in [0,x_M]$ by the maximality of x_M. Based on Lemma 4, the first t entries are bounded by $(x^U_{t,1},\ldots,x^U_{t,t})$ and $(x^L_{t,1},\ldots,x^L_{t,t})$. However, we have seen that $(x^U_{t,1},\ldots,x^U_{t,\tau}) = (x^L_{t,1},\ldots,x^L_{t,\tau})$, thus the first τ entries of its solution are equal to $(x^U_{t,1},\ldots,x^U_{t,\tau})$ no matter what the future is. ∎



Minghong Lin received the B.Sc. degree in Computer Science from University of Science and Technology of China in 2006, and the M.Phil degree in Computer Science from the Chinese University of Hong Kong in 2008. Currently he is a Ph.D. student in Computer Science at California Institute of Technology. His research interests include energy efficient computing, online algorithms and nonlinear optimization. He has received best paper award at IEEE INFOCOM'11, IGCC'12 and best student paper award at ACM GREENMETRICS'11.



Adam Wierman is an Assistant Professor in the Department of Computing and Mathematical Sciences at the California Institute of Technology, where he is a member of the Rigorous Systems Research Group (RSRG). He received his Ph.D., M.Sc. and B.Sc. in Computer Science from Carnegie Mellon University in 2007, 2004, and 2001, respectively. He received the ACM SIGMETRICS Rising Star award in 2011, and has also received best paper awards at ACM SIGMETRICS, IFIP Performance, IEEE INFOCOM, and ACM GREENMETRICS. He has also received multiple teaching awards, including the Associated Students of the California Institute of Technology (ASCIT) Teaching Award. His research interests center around resource allocation and scheduling decisions in computer systems and services. More specifically, his work focuses both on developing analytic techniques in stochastic modeling, queueing theory, scheduling theory, and game theory, and applying these techniques to application domains such as energy-efficient computing, data centers, social networks, and the electricity grid.

Lachlan Andrew (M'97-SM'05) received the B.Sc., B.E. and Ph.D. degrees in 1992, 1993, and 1997, from the University of Melbourne, Australia. Since 2008, he has been an associate professor at Swinburne University of Technology, Australia, and since 2010 he has been an ARC Future Fellow. From 2005 to 2008, he was a senior research engineer in the Department of Computer Science at Caltech. Prior to that, he was a senior research fellow at the University of Melbourne and a lecturer at RMIT, Australia. His research interests include energy-efficient networking and performance analysis of resource allocation algorithms. He was co-recipient of the best paper award at IGCC2012, IEEE INFOCOM 2011 and IEEE MASS 2007. He is a member of the ACM.

Eno Thereska received his PhD (ECE, 2007), Masters (ECE, 2003) and Bachelor (ECE + CS + Math minor, 2002) degrees all from Carnegie Mellon. He's been a Researcher with Microsoft Research since 2007. He has broad interests in systems but also enjoys venturing into other areas occasionally, like machine learning and HCI. He was co-recipient of the best paper award at IEEE INFOCOM 2011 and Usenix FAST 2005, and the best student paper at Usenix FAST 2004.