Energy-Efficient Dynamic Capacity Provisioning in Server Farms

Anshul Gandhi[*] **Varun Gupta**[*]
Mor Harchol-Balter[*] **Michael Kozuch**[†]

March 2010
CMU-CS-10-108

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

[*]School of Computer Science, Carnegie Mellon University
[†]Intel Research, Pittsburgh

Research supported by an NSF CCR-0615262 grant.

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of NSF.

Keywords: Server farms, Power management, Queueing theory

Abstract

A central question in designing server farms today is how to efficiently provision the number of servers to handle unpredictable demand patterns, so as to extract the best performance while not wasting energy. While one would like to turn servers off when they become idle to save energy, the large setup cost (both, in terms of setup time and energy penalty) needed to switch the server back on can adversely affect performance. The problem is made more complex by the fact that today's servers provide multiple sleep or standby states which trade off the setup cost with the power consumed while the server is 'sleeping'. With so many controls, finding the optimal server pool management policy is an almost intractable problem – How many servers should be *on* at any given time, how many should be *off*, and how many should be in some *sleep* state?

In this paper, we analyze server farm management policies, and prove that it suffices to consider a small, natural class of policies to guarantee near-optimal behavior. We employ the popular metric of Energy-Response time Product (ERP) to capture the energy-performance tradeoff, and present the first theoretical results on the optimality of server farm policies under both stationary and time-varying demand patterns. We find that there exists a very small set of natural policies that always contains the optimal policy for a single server, and a near-optimal policy for multi-server systems, for a stationary demand pattern. We also consider server farm management policies for time-varying demand patterns and propose a robust, traffic-oblivious policy which is asymptotically optimal as the load becomes large.

1 Introduction

Motivation

Server farm power consumption accounts for more than 1.5% of the total electricity usage in the U.S., at a cost of nearly \$4.5 billion [23]. The rising cost of energy and the tremendous growth of data centers will result in even more spending on power consumption. Unfortunately, only 20-30% of the total server capacity is used on average [8]. The main culprit of this wastage are idle servers in over-provisioned server farms.

Idle servers consume about 60% of their peak power [8]. While a lot of energy can be saved by turning *idle* servers *off*, turning on an *off* server incurs a significant cost. The *setup cost* takes the form of both a time delay, which we refer to as the *setup time*, and an *energy penalty*. Thus it is not obvious whether *idle* servers should be turned *off*. Another option is to put *idle* servers into some *sleep* state. While a server in *sleep* mode consumes more power than an *off* server, the setup cost for a sleeping server is lower than that for an *off* server. Today's state-of-the-art servers come with an array of *sleep* states, leaving it up to the server farm manager to determine which of these is best.

Goal and metric

There is a clear tradeoff between leaving *idle* servers on, and thus minimizing mean response time, versus turning *idle* servers *off* (or putting them to *sleep*), which hurts response time but may save power. Optimizing this tradeoff is a difficult problem, since there are an infinite number of possible server farm management policies. Our goal in this paper is to find a simple class of server farm management policies, which optimize (or nearly optimize) the above tradeoff. We also seek simple rules of thumb that allow designers to choose from this class of near-optimal policies. In doing so, we greatly simplify the job of the server farm manager by reducing the search space of policies that he/she needs to choose from.

To capture the tradeoff involved in energy and performance, and to compare different policies, we use the Energy-Response time Product (ERP) metric, also known as the Energy-Delay Product (EDP) [18, 22, 11, 19, 17]. For a control policy π, the ERP is given by:

$$ERP^{\pi} = \mathbf{E}[P^{\pi}] \cdot \mathbf{E}[T^{\pi}]$$

where $\mathbf{E}[P^{\pi}]$ is the long-run average power consumed under the control policy π, and $\mathbf{E}[T^{\pi}]$ is mean customer response time under policy π. Minimizing the quantity ERP can be seen as maximizing the "performance-per-watt", with performance being defined as the inverse of mean response time. While ERP is widely accepted as a suitable metric to capture energy-performance tradeoffs, we believe we are the first to analytically address optimizing the metric of ERP in server farms.

Note that there are other performance metrics that also capture the tradeoff between response time and energy, for example, a weighted sum of the mean response time and mean power (ERWS) [3, 4, 24]. However, the ERWS metric implies that a reduction in mean response time from 1001 sec to 1000 sec is of the same value as a reduction from 2 sec to 1 sec. By contrast, the ERP implies that a reduction in mean response time from 2 sec to 1 sec is much better than a reduction from 1001 sec to 1000 sec, which is more realistic. One reason for the popularity of ERWS is that it is a nicer metric to handle analytically, being a single expectation, and hence additive over time. Therefore, one can optimize the ERWS metric via Markov Decision Processes, for example. From the point of view of worst case sample path based analysis, this metric allows comparing arbitrary policies to the optimal policy via potential function arguments [15]. However, ERP, being a product of two expectations, does not allow a similar analysis. Other realistic metrics of interest include minimizing total energy given bounds on, say, the 95%tile of response times.

Summary of Contributions

We consider a specific set of server farm management policies (defined in Table 1) and prove that it contains the optimal policy for the case of a single server, and also contains a near-optimal policy for the case of multi-server systems, assuming a stationary demand pattern. For the case of time-varying demand patterns, we develop a traffic-oblivious policy that can auto-scale the server farm capacity to adapt to the incoming load, and prove that this policy is optimal as the load approaches infinity. Throughout this paper, for analytical tractability, we make the assumption of Exponentially distributed job sizes and a Poisson arrival process. However, the setup time distribution is assumed to be Deterministic. We formally define the traffic model and the model for servers' *sleep* state dynamics in Section 3.

- We begin in Section 4 by considering a single-server system. The arrival process is Poisson with a known mean arrival rate. There is an infinite range of policies that one could consider for managing a single server, for example, when the server goes *idle*, one could immediately turn it *off* (INSTANTOFF), or alternatively, move the server to a specific *sleep* state (SLEEP). One could also just leave the server *idle* when it has no work to do (NEVEROFF). Another possibility is to turn an *idle* server *off* with some probability p, and leave it *idle* with probability $(1 - p)$. One could also delay turning on an *off* server until a certain number of jobs have accumulated in the queue. Also, when turning on an *off* server, one could transition through *sleep* states, with each successive transition moving the server closer to the *on* state. Within this wide range of policies, we prove that one of the policies, NEVEROFF, INSTANTOFF or SLEEP, is always optimal. Refer to Table 1 for the exact definitions of these policies.

- In Section 5, we consider the case of multi-server systems. The arrival process is Poisson with a known mean arrival rate. We assume that there are enough servers so that we are not

Policy	**Single-Server**	**Multi-Server**
NEVEROFF	Whenever the server goes *idle*, it remains *idle* until a job arrives.	A fixed optimally chosen number n^* (with respect to ERP) of servers are maintained in the *on* or *idle* states. If an arrival finds a server *idle*, it starts serving on the *idle* server. Arrivals that find all n^* servers *on* (busy) join a central queue from which servers pick jobs when they become *idle*.
INSTANTOFF	Whenever the server goes *idle*, it turns *off*. It remains *off* until there is no work to process, and begins to turn *on* as soon as work arrives.	Whenever a server goes *idle*, and there are no jobs in the queue, the server turns *off*. Otherwise it picks a job from the queue to serve. At any moment in time, there are some number of servers that are *on* (busy), and some number of servers that are in *setup*. Every arrival puts a server into *setup* mode, unless the number of servers in *setup* already exceeds the number of jobs in the queue. A job does not necessarily wait for the full setup time since it can be run on a different server that becomes free before the setup time is complete, leaving its initially designated server in *setup*.
SLEEP(S)	Whenever a server goes *idle*, it goes into the *sleep* state S. It remains in *sleep* state S until there is no work to process, and begins to wake up as soon as work arrives.	A fixed optimally chosen number n^* of servers are maintained in the *on*, *off* or *sleep* states. Whenever a server goes *idle*, and there are no jobs in the queue, it goes into the *sleep* state S. Otherwise it picks a job from the queue to serve. Every arrival wakes a sleeping server and puts it into *setup*, unless the number of servers in *setup* already exceeds the number of jobs in the queue.

Table 1: A summary of the different policies considered in this paper, and their description in the single-server and multi-server cases.

constrained by the available capacity. In the multi-server setting, we have an even wider range of policies to choose from. For example, some servers could be turned *off* when *idle*, some could be moved to a specific *sleep* state, and the rest may be kept *idle*. Another possibility is to turn *idle* servers *off* based on the state of the system, such as the total number of *idle* servers, or the total number of *off* servers, etc. One could also delay turning on an *off* server until a certain number of jobs have accumulated in the queue, or delay turning *off* an *idle* server until some time has elapsed. Clearly, there is a huge set of admissible policies. Here again, we show that one of NEVEROFF, INSTANTOFF or SLEEP (defined in Table 1 for a multi-server system) is near-optimal.

- In Section 6 we consider a time-varying arrival pattern with the aim of finding policies which can auto-scale the capacity while being oblivious to the traffic intensity. This situation is even more complicated than in Section 5, since a server farm management policy might now also take into account the history of arrivals or some predictions about the future arrivals. The class of available policies is now even larger. For the time-varying case, we introduce a new policy DELAYEDOFF. Under the DELAYEDOFF policy, a server is only turned *off* if it does not receive any jobs to serve in time t_{wait}. If an arrival finds more than one server *idle* on arrival, it is routed to the server which was *most recently busy* (MRB). Otherwise, the arriving job turns *on* an *off* server.

The MRB routing proposed above turns out to be crucial for the near-optimality of DELAYEDOFF. Intuitively, MRB routing increases the variance of the idle periods of the servers when compared to random or round-robin routing, and yields the property that the longer a server has been idle, the longer it is likely to stay idle. We prove that DELAYEDOFF is asymptotically optimal as the load becomes large. Policies similar to DELAYEDOFF have been proposed in the literature but applied to individual devices [15, 10, 21], whereas in our case we propose to apply it to a pool of homogeneous interchangeable servers under MRB routing. We provide both analytical and simulation evidence in favor of the auto-scaling capabilities of DELAYEDOFF and show that it compares favorably to an offline, traffic-aware capacity provisioning policy.

2 Prior work

Prior analytical work in server farm management to optimize energy-performance tradeoff can be divided into *stochastic analysis*, which deals with minimizing average power/delay or the tail of power/delay under some probabilistic assumptions on the arrival sequence, and *worst-case analysis*, which deals with minimizing the cost of worst-case arrival sequences.

Stochastic Analysis

The problem of server farm management is very similar in flavor to two well studied problems in the stochastic analysis community: operator staffing in call centers and inventory management. In call center staffing, the servers are operators, who require a salary (power) when they are working. Similarly to our problem, these operators require a setup cost to bring an employee into work, however, importantly, all analysis in call center staffing has ignored this setup cost.

The operator staffing problem involves finding the number of operators (servers) which minimize a weighted sum of delay costs experienced by users and the monetary cost of staffing operators. While this problem has received significant attention under the assumption of stationary (non-time-varying) demand (see [9] for recent results), there is significantly less work for the time-varying case, one exception being [16]. In [16], the authors consider the problem of dynamic staffing based on knowing the demand pattern so as to maintain a target probability of a user finding all servers busy on arrival.

Within inventory management, the problem of capacity provisioning takes the form: how much inventory should one maintain so as to minimize the total cost of unused inventory (holding cost, in our case *idle* power) and waiting cost experienced by orders when there is no inventory in stock (queueing delay of users). Conceptually this problem is remarkably similar to the problem we consider, and the two common solution strategies employed, known as Make to Order and Make to Stock, are similar in flavor to what we call INSTANTOFF and NEVEROFF, respectively (see [2], for example). However, in our case servers can be turned *on* in parallel, while in inventory management it is assumed that inventory is produced *sequentially* (this is similar to allowing at most one server to be in *setup* at any time).

Worst-case Analysis

The theoretical CS community has been interested in power management from the point of view of minimizing worst case cost, for example ERWS (See [14] for a recent survey). Again, none of the prior work encompasses a setup time and is more applicable to a single device than a server farm. The performance metrics used are also very different from ERP.

The work can primarily be split in terms of results on speed scaling algorithms, and results on algorithms for powering down devices. In the realm of speed scaling, the problem flavors considered have been minimizing energy or maximum temperature while meeting job deadlines [25, 6, 7], minimizing mean response time subject to a bound on total energy [20], and minimizing the ERWS [5, 24]. However, again all these papers assume that the speed level can be switched *without any setup costs*, and hence are mainly applicable to single stand-alone devices, since in multi-server systems setup costs are required to increase capacity.

The work on powering down devices is more relevant to the problem we consider, and due to sample path guarantees, these results naturally lead to traffic-oblivious powering down schemes. In [15] the authors consider the problem of minimizing total energy consumed under the constraint that a device must instantly turn on when a job arrives. Further, [15] assumes that there is *no setup time* while turning on a device, only an energy penalty.

3 Model

Figure 1 illustrates our server farm model. We assume n homogeneous servers, where each server can process any job, and thus the servers are interchangeable. Jobs arrive from outside the system, to a central queue, according to a Poisson process. In Sections 4 and 5, we consider a fixed arrival rate, λ. However, in Section 6, we consider a time-varying arrival rate, $\lambda(t)$. We assume the job sizes are independent and identically distributed according to an Exponentially distributed random variable S, with rate μ. The quantity $\rho(t) = \lambda(t) \cdot \mathbf{E}[S]$ is used to denote the instantaneous load, or the rate at which work is entering the system at time t. In Sections 4 and 5, where we assume $\lambda(t) = \lambda$, we have $\rho = \lambda\mathbf{E}[S]$. In the case of a multi-server system with n servers, $0 \leq \rho < n$. Here ρ represents

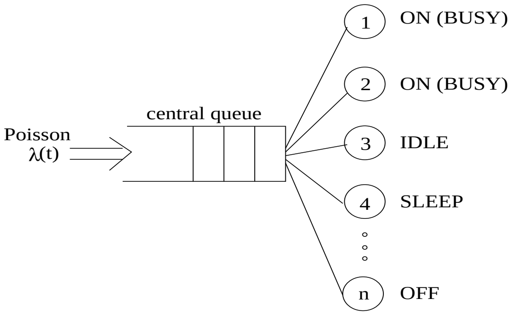


Figure 1: Illustration of our server farm model.

the minimum number of servers needed to maintain a stable system.

Each server can be in one of the following states: *on* (busy)[1], *idle*, *off*, or any one of $N-1$ *sleep* states: S_1, S_2, …, S_{N-1}. For convenience, we sometimes refer to the *idle* state as S_0 and the *off* state as S_N. The associated power values are P_{ON}, $P_{IDLE} = P_{S_0}$, P_{S_1}, …, $P_{S_N} = P_{OFF}$. We shall assume the ordering $P_{ON} > P_{IDLE} > P_{S_1} > \ldots > P_{S_{N-1}} > P_{OFF} = 0$. The server can only serve jobs in the *on* state [2]. The time to transition from initial state, S_i, to final state, S_f, is denoted by $T_{S_i \to S_f}$ and is a constant (not a random variable). Rather obviously, we assume $T_{ON \to IDLE} = T_{IDLE \to ON} = 0$. Further, the average power consumed while transitioning from state S_i to S_f is given by $P_{S_i \to S_f}$.

Model Assumptions: For analytical tractability, we will relax the above model a little. We will assume that the time to transition from a state to any state with lower power is zero. Therefore, $T_{ON \to OFF} = T_{S_i \to OFF} = 0$, for all i. This assumption is justified because the time to transition back to a higher power state is generally considerably larger than the time to transition to the lower power state, and hence dominates the performance penalties. Further, we will assume that the time to transition from a state S_i to any higher power state is only dependent on the low power state, and we will denote this simply as T_{S_i}. Therefore, $T_{OFF \to IDLE} = T_{OFF \to S_i} = T_{OFF}$, for all i. Note that $0 = T_{IDLE} < T_{S_1} < \ldots < T_{S_{N-1}} < T_{OFF}$. This assumption is justified because in current implementations there is no way to go between two *sleep* states without first transitioning through the $IDLE$ state. Regarding power usage, we assume that when transitioning from a lower power state, S_i, to a higher power state S_f, we consume power $P_{S_i \to S_f} = P_{ON}$.

The results of this paper are derived under the *Model Assumptions.* We have validated these assumptions within an experimental data center in our lab.

[1] We use italicized *on* to denote the state when the server is busy, and without italics when we are colloquially referring to either the busy or idle state.

[2] P_{ON} need not necessarily denote the peak power at which a job is served, but is used as a proxy for the average power consumed during the service of a job. Indeed, while applying our model, we would first profile the workload to measure the average power consumed during a job's execution, and use it as P_{ON}.

3.1 Simulation methodology

We use a discrete event simulator written in the C++ language to verify our theoretical results for the various dynamic capacity provisioning policies used in the paper. Our simulator models a server farm based on the above *Model Assumptions*.

Throughout the paper, we use simulation results based on the following server characteristics: $T_{OFF} = 200s$, $T_{SLEEP} = 60s$, $P_{OFF} = 0W$, $P_{SLEEP} = 10W$, $P_{IDLE} = 150W$ and $P_{ON} = 240W$. These parameter values are based on measurements for the Intel Xeon E5320 server, running the CPU-bound LINPACK [13] workload.

4 Optimal Single Server policies

As the first step towards our goal of finding policies for efficiently managing server pools, we begin by analyzing the case of a single server system. Recall that our aim is to find the policy that minimizes ERP under a Poisson arrival process of known intensity. Theorem 1 below states that for a single server, the optimal policy is included in the set {NEVEROFF, INSTANTOFF, SLEEP} (defined in Section 1), and hence there is no need to consider any other capacity provisioning policy.

Theorem 1 *For the single server model with a Poisson(λ) arrival process and i.i.d. Exponentially distributed job sizes, the optimal policy for minimizing ERP is one of* NEVEROFF*,* INSTANTOFF *or* SLEEP(S)*, where S is the optimally chosen* sleep *state among the existing* sleep *states.*

Before we prove Theorem 1, we would like to point out that this is quite a non-intuitive result, and in general we do not expect it to hold for other metrics such as ERWS. The theorem rules out a large class of policies, for example those which may randomize between transitioning to different *sleep* states, or policies which move from one *sleep* state to another, or those which may wait for a few jobs to accumulate before transitioning to the *on* state. While ERP, being a product of expectations, is a difficult metric to address analytically, for the single-server case we are able to obtain tight optimality results by deriving explicit expressions for ERP.

Proof of Theorem 1: We give a high-level sketch of the proof in terms of four lemmas, whose proofs are deferred to Appendix A. These lemmas successively narrow down the class of optimal policies, until we are left with only NEVEROFF, INSTANTOFF and SLEEP.

Definition 1 *Let Π_{mixed} denote the class of randomized policies whereby a server immediately transitions to power state S_i ($i \in \{0, \ldots, N\}$) with probability p_i on becoming* idle*. Given that the server went into power state S_i, it stays in S_i until N_i jobs accumulate in the queue, where $N_i \geq 1$ is a random variable with $q_{ij} = \mathbf{Pr}[N_i = j]$. Once N_i jobs have accumulated, the server immediately begins transitioning to the* on *state, and stays there until going* idle*.*

Lemma 1 *The optimal policy lies in the set Π_{mixed}.*

Lemma 2 *Consider a policy $\pi \in \Pi_{mixed}$ with parameters as in Definition 1. The mean response time for policy π under a Poisson(λ) arrival process with i.i.d.* $\mathrm{Exp}(\mu)$ *job sizes is given by:*

$$\mathbf{E}[T] = \frac{\sum_{i=0}^{N} p_i \sum_{j=1}^{\infty} q_{ij} r_{ij}}{\sum_{i=0}^{N} p_i \sum_{j=1}^{\infty} q_{ij} \left(j + \lambda T_{S_i}\right)} \tag{1}$$

where,

$$r_{ij} = \frac{j + \lambda T_{S_i}}{\mu - \lambda} + \left[j T_{S_i} + \frac{j(j-1)}{2\lambda} + \frac{\lambda T_{S_i}^2}{2} \right] \tag{2}$$

and the average power for policy π is given by:

$$\mathbf{E}[P] = \frac{\sum_{i=0}^{N} p_i \sum_{j=1}^{\infty} q_{ij} \left(j(\rho P_{ON} + (1-\rho) P_{S_i}) + \lambda T_{S_i} P_{ON}\right)}{\sum_{i=0}^{N} p_i \sum_{j=1}^{\infty} q_{ij} \left(j + \lambda T_{S_i}\right)}. \tag{3}$$

Lemma 3 *The optimal strategy for a single server must be pure. That is, $p_i = 1$ for some $i \in \{0, \ldots, N\}$, and $N_i \equiv n_i$ for some integer $n_i \geq 1$.*

Lemma 4 *The optimal pure strategy dictates that $n_i = 1$, if the optimal* sleep *state is S_i.*

Lemma 1 is proved using a sample path argument and crucially depends on the Poisson arrival process and the *Model Assumptions* for the *sleep* states of the server, and in fact holds for any metric that is increasing in mean response time and mean power. Lemma 3 relies on the structure of ERP metric. While Lemma 3 also holds for the ERWS metric (with a much simpler proof), it does not necessarily hold for general metrics such as the product of the mean power and the square of the mean response time. Lemma 4 also relies on the structure of the ERP metric and does not hold for other metrics such as ERWS. ■

Lemma 5 *Assuming a Poisson(λ) arrival process, and* $\mathrm{Exp}(\mu)$ *job sizes, the mean response time and mean power for NEVEROFF, INSTANTOFF and SLEEP are given by:*

$$\mathbf{E}[T] = \frac{1}{\mu - \lambda} + \frac{T_{S_i}(1 + \lambda T_{S_i}/2)}{1 + \lambda T_{S_i}} \tag{4}$$

$$\mathbf{E}[P] = \frac{\rho P_{ON} + (1-\rho) P_{S_i} + \lambda T_{S_i} P_{ON}}{1 + \lambda T_{S_i}} \tag{5}$$

where $S_i = IDLE$ for NEVEROFF, $S_i = OFF$ for INSTANTOFF, and S_i is the sleep *state that we transition to in SLEEP.*

Proof: Follows as a consequence of Lemma 2. ■

Lemma 5 provides expressions for the mean response time and the mean power consumption under the NEVEROFF, INSTANTOFF and SLEEP policies. These expressions allow us to determine regimes of load and mean job sizes for which each policy is best, with respect to ERP. Although not shown (for lack of space), we find that NEVEROFF is typically superior to the other policies, unless the load is low *and* the mean job size is high, resulting in very long idle periods. In that latter case, INSTANTOFF or one of the SLEEP policies is superior, where the choice between these depends on the parameters of the *sleep* and *off* states. Eqs. (4) and (5) are also helpful for guiding a server architect towards designing useful *sleep* states, because they enable the evaluation of ERP for each candidate *sleep* state.

5 Near-Optimal Multi-server policies

In this section, we extend our results for single server systems to the multi-server systems with a fixed known arrival rate, with the goal of minimizing ERP. Inspired by the results in Section 4, where we found the best of NEVEROFF, INSTANTOFF and SLEEP to be the optimal policy, we intuit that in the multi-server case, one of NEVEROFF, INSTANTOFF and SLEEP will be close to optimal as well. We make this intuition precise in Section 5.1, and in Section 5.2, we provide simple guidelines for choosing the right policy from among this set, depending on the system parameters.

5.1 Near-optimality results

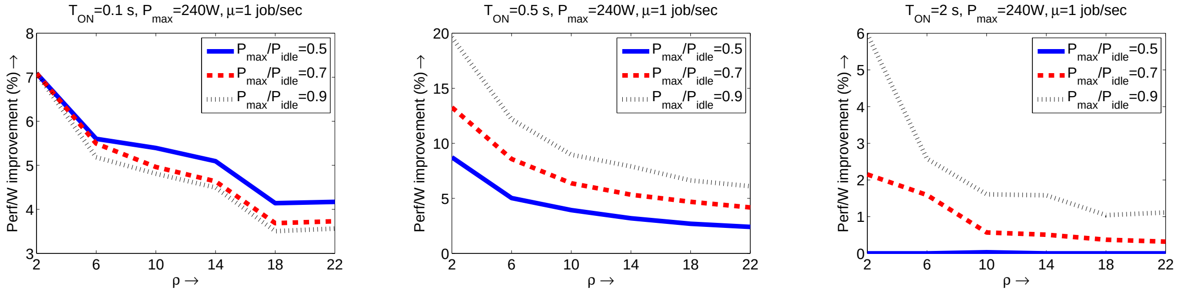
Theorem 2 *Let Π_{OFF} denote the class of policies which only involve the states* on*,* idle *and* off*. The ERP of the best of NEVEROFF and INSTANTOFF is within 20% of the ERP of the optimal policy in Π_{OFF} when $\rho \geq 10$. When $\rho \geq 20$, the performance gap is smaller than 12%.*

Theorem 3 *Let Π_{S_i} denote the class of policies which only involve the states* on*,* idle *and the S_i* sleep *state. For arbitrary S_i (that is P_{S_i} and T_{S_i}), the ERP of the best of NEVEROFF and SLEEP with* sleep *state S_i is within 30% of the ERP of the optimal policy in Π_{S_i} when $\rho \geq 10$. When $\rho \geq 20$, the performance gap is smaller than 23%.*

We defer the proof of Theorem 2 to Appendix B. The proof of Theorem 3 is similar to the proof of Theorem 2, and is also presented in Appendix C.

The main idea behind proving Theorems 2 and 3 is getting a reasonably good lower bound on the ERP for the optimal policy. We believe that in reality, the simple NEVEROFF, INSTANTOFF, and SLEEP policies are better than our Theorems suggest. To justify this conjecture we perform the following simulation experiment. We focus on the case in Theorem 2 of policies involving *on*, *idle* and *off* states. Note that as we mentioned earlier, due to the metric of ERP, we can not utilize the framework of Markov Decision Processes/Stochastic Dynamic Programming to numerically obtain the optimal policy. Instead we limit ourselves to the following class of threshold policies:

THRESHOLD(n_1, n_2): At least n_1 servers are always maintained in *on* or *idle* state. If an arrival finds a server *idle*, it begins service. If the arrival finds all servers *on* (busy) or turning on, but this number is less than $n_2 \geq n_1$, then the arrival turns on an *off* server. Otherwise the arrival waits in a queue. If a server becomes *idle* and the queue is empty, the server turns *off* if there are at least n_1 other servers which are *on*.



(a) $T_{OFF} = 0.1\mathbf{E}[S]$ (b) $T_{OFF} = 0.5\mathbf{E}[S]$ (c) $T_{OFF} = 2\mathbf{E}[S]$

Figure 2: Comparison of the performance of THRESHOLD policy against the best of NEVEROFF and INSTANTOFF policies. The y-axis shows the percentage improvement in ERP afforded by the THRESHOLD policy.

The THRESHOLD policy can be seen as a mixture of NEVEROFF with n_1 servers, and INSTANTOFF with $(n_2 - n_1)$ servers. Therefore, THRESHOLD represents a broad class of policies (since n_1 and n_2 can be set arbitrarily), which includes NEVEROFF and INSTANTOFF. In Figure 2, we show the improvement in ERP afforded by the optimal THRESHOLD policy over the best of NEVEROFF and INSTANTOFF for various values of ρ, T_{OFF} and $\frac{P_{IDLE}}{P_{ON}}$. We see that if T_{OFF} is small (Figure 2 (a)), the ERP gain of the THRESHOLD policy over the best of NEVEROFF and INSTANTOFF is marginal ($< 7\%$). This is because in this case, INSTANTOFF is close to optimal. At the other end, when T_{OFF} is large (Figure 2 (c)), the ERP gain of the THRESHOLD policy over the best of NEVEROFF and INSTANTOFF are again marginal ($< 6\%$), because now NEVEROFF is close to optimal. We expect the optimal THRESHOLD policy to outperform the best of NEVEROFF and INSTANTOFF when T_{OFF} is moderate (comparable to $\frac{P_{IDLE} \cdot \mathbf{E}[S]}{P_{ON}}$). In Figure 2 (b), we see that this is indeed the case. However, the gains are still moderate (an improvement of 10% when $\rho \geq 10$ and at most 7% when $\rho \geq 20$ when P_{IDLE} is high).

5.2 Choosing the right policy

Based on the results of Section 5.1, to provision a multi-server system with a fixed known arrival rate, it suffices to only consider the policies NEVEROFF, INSTANTOFF and SLEEP. Below we develop a series of simple rules of thumb that help a practitioner choose between these policies.

We start by deriving the n^* (optimal number of servers) for NEVEROFF and SLEEP. This is done in Rules of Thumb #1 and #2, respectively. Then, we use these n^* values to derive a criterion that a practitioner can use to choose between NEVEROFF, INSTANTOFF and SLEEP. This is done in Rule of Thumb #3.

Define $\beta^*(\cdot)$ to be the following function:

$$\beta^*(x) = \text{arg min}_{\beta>0}\left(\frac{\alpha(\beta)}{\beta} + \beta\cdot x\right) \tag{6}$$

where,

$$\alpha(\beta) = \left[1 + \sqrt{2\pi}\beta\Phi(\beta)e^{\frac{\beta^2}{2}}\right]^{-1} \tag{7}$$

and $\Phi(\cdot)$ is the c.d.f. of a standard Normal variate.

Rule of Thumb #1: Choosing n^* for NEVEROFF
For the parameter regime where NEVEROFF is the chosen policy,

$$n^* \approx \rho + \beta^*(P_{IDLE}/P_{ON})\sqrt{\rho} \tag{8}$$

and $\beta^*(\cdot)$ is given by (6).

Rule of Thumb #2: Choosing n^* for SLEEP
For the parameter regime where SLEEP with *sleep* state S_i is the chosen policy,

$$n^* \approx \rho' + \beta^*(P_{S_i}/P_{ON})\sqrt{\rho'} \tag{9}$$

where $\rho' = \rho\left(1 + \frac{T_{S_i}}{\mathbf{E}[S]}\right)$and $\beta^*(\cdot)$ is given by (6).

We now justify Rule of Thumb #1, which is popular in the call center staffing literature as the 'square-root staffing rule'. We begin with the following lemma from [12]:

Lemma 6 (Halfin and Whitt [12]) *Consider a sequence of $M/M/s_n$ systems with load ρ_n in the nth system. Let α_n denote the probability that an average customer finds all servers busy in the nth system. Then,*

$$\lim_{\rho_n\to\infty} \alpha_n = \alpha(\beta) \textit{ if and only if } \lim_{\rho_n\to\infty}\frac{s_n - \rho_n}{\sqrt{\rho_n}} = \beta. \tag{10}$$

In this case, $\alpha(\beta)$ is given by Eq. (7), and the mean number of jobs in the nth system, $\mathbf{E}\left[N^{M/M/s_n}\right]$, *satisfies:*

$$\lim_{\rho_n \to \infty} \frac{\mathbf{E}\left[N^{M/M/s_n}\right] - \rho_n}{\sqrt{\rho_n}} = \frac{\alpha(\beta)}{\beta}. \tag{11}$$

Lemma 7 *For the* NEVEROFF *policy, square root staffing with $n^* = \rho + \beta^*(P_{IDLE}/P_{ON})\sqrt{\rho}$ is asymptotically optimal for minimizing the metric of ERP.*

Proof: Consider a sequence of $M/M/s_n$ systems with load ρ_n in the nth system. Let $s_n \sim \rho + g(\rho_n) + o(g(\rho_n))$. From [12], we have that $\mathbf{E}\left[N^{M/M/s_n}\right] \sim \rho_n + \frac{\rho_n}{g(\rho_n)}\alpha_n$ where α_n denotes the stationary probability that all s_n servers are busy in the nth system. Also, $\mathbf{E}\left[P^{M/M/s_n}\right] \sim \rho P_{ON} + g(\rho_n)P_{IDLE}$, which gives

$$\mathbf{E}\left[N^{M/M/s_n}\right] \cdot \mathbf{E}\left[P^{M/M/s_n}\right] \sim \rho_n^2 P_{ON}\left(1 + \frac{\alpha_n}{g(\rho_n)} + \frac{g(\rho_n)}{\rho_n}\frac{P_{IDLE}}{P_{ON}} + o() \text{ terms.}\right)$$

Since $\alpha_n \to 0$ when $g(\rho_n) = \omega(\sqrt{\rho_n})$, and $\alpha_n \to 1$ when $g(\rho_n) = o(\sqrt{\rho_n})$, it is easy to see that the optimal choice is $g(\rho_n) = \beta\sqrt{\rho_n}$ for some constant β. This yields:

$$ERP^{NEVEROFF} \quad \sim \quad \rho_n \mathbf{E}[S] P_{ON}\left(1 + \frac{\frac{\alpha(\beta)}{\beta} + \beta\frac{P_{IDLE}}{P_{ON}}}{\sqrt{\rho_n}}\right) \tag{12}$$

Optimizing the above yields the expression for β^*. ■

The square-root staffing rule is also called the Quality and Efficiency Driven regime, because it balances the sub-optimality in the performance (Quality) and resource utilization (Efficiency), both being $\Theta\left(\frac{1}{\sqrt{\rho}}\right)$, and hence optimizes the ERWS metric. Here we have shown that square-root staffing also optimizes the ERP metric, albeit with a different β.

The justification for Rule of Thumb #2 is along the same lines. We expect the SLEEP(S$_i$) policy to outperform NEVEROFF when T_{S_i} is small enough so that almost all jobs turn on a *sleeping* server and get served there. This is equivalent to an $M/G/\infty$ system with $G \sim S + T_{S_i}$. However, since $P_{S_i} > 0$, we optimize the number of servers by following Rule of Thumb #1, but with mean job size replaced by $\mathbf{E}[S] + T_{S_i}$, or equivalently $\rho' \leftarrow \rho\left(1 + \frac{T_{S_i}}{\mathbf{E}[S]}\right)$, and $P_{IDLE} \leftarrow P_{S_i}$. This gives us:

$$ERP^{SLEEP(S_i)} \quad \sim \quad \rho\mathbf{E}[S]\left(1 + \frac{T_{S_i}}{\mathbf{E}[S]}\right)^2 P_{ON}\left(1 + \frac{\frac{\alpha(\beta)}{\beta} + \beta\frac{P_{S_i}}{P_{ON}}}{\sqrt{\rho\left(1 + \frac{T_{S_i}}{\mathbf{E}[S]}\right)}}\right) \tag{13}$$

We now provide a Rule of Thumb to choose between the NEVEROFF, INSTANTOFF and SLEEP policies, by proposing approximations for their ERP. We expect the INSTANTOFF policy to outperform NEVEROFF and SLEEP when T_{OFF} is small enough compared to $\mathbf{E}[S]$, so that the penalty to turn on an *off* server is negligible compared to the necessary cost of serving the job. In this regime, we can approximate the ERP of INSTANTOFF by $ERP^{INSTANTOFF} \approx \lambda P_{ON} \left(\mathbf{E}[S] + T_{OFF}\right)^2$, which is an upper bound obtained by forcing every job to run on the server that it chooses to turn on on arrival. The ERP of NEVEROFF with optimal number of servers is approximated by Eq. (12), with $\rho_n = \rho$ and $\beta = \beta^*(P_{IDLE}/P_{ON})$. For SLEEP, we again expect SLEEP(S_i) policy to outperform NEVEROFF when T_{S_i} is small enough so that almost all jobs turn on a *sleeping* server and get served there. In this regime, we can approximate the ERP of SLEEP by Eq. (13), with $\beta = \beta^*(P_{S_i}/P_{ON})$. Using the above approximations for ERP, we can choose between the INSTANTOFF, NEVEROFF and SLEEP policies:

Rule of Thumb #3: Which policy to use?
We first associate each policy with an index. The index for INSTANTOFF is given by $\left(1 + \frac{T_{OFF}}{\mathbf{E}[S]}\right)^2$. The index for NEVEROFF is given by $\left(1 + \frac{\gamma(P_{IDLE}/P_{ON})}{\sqrt{\rho}}\right)$, and for SLEEP with state S_i by $\left(1 + \frac{T_{S_i}}{\mathbf{E}[S]}\right)^2 \left(1 + \frac{\gamma(P_{S_i}/P_{ON})}{\sqrt{\rho\left(1+\frac{T_{S_i}}{\mathbf{E}[S]}\right)}}\right)$. The function $\gamma(\cdot)$ is given by

$$\gamma(x) = \min_{\beta>0} \left(\frac{\alpha(\beta)}{\beta} + \beta \cdot x\right) \tag{14}$$

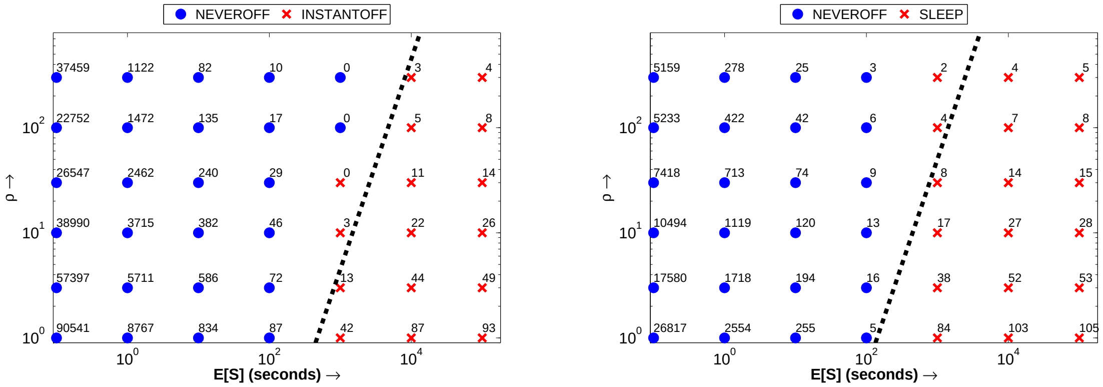
with $\alpha(\beta)$ given by (7). Finally, the policy with the smallest index is chosen.

If we compare INSTANTOFF and NEVEROFF, Rule of Thumb #3 says that if T_{OFF} is sufficiently small compared to $\mathbf{E}[S]$ and $\frac{1}{\sqrt{\rho}}$, then one should choose INSTANTOFF. Figure 3(a) verifies the accuracy of the above rule of thumb. Observe that in the region where our rule of thumb mispredicts the better policy, the gains of choosing either policy over the other are minimal. Similarly, the dashed line in Figure 3(b) indicates that the theoretically predicted split between the NEVEROFF and SLEEP policies is in excellent agreement with simulations.

6 A Traffic-oblivious dynamic capacity provisioning policy

Thus far we have considered a stationary demand pattern. Our goal in this section is to propose a server farm management policy with near-optimal ERP when the demand pattern is time-varying and unknown. Specifically we assume that the arrival process is Poisson, with unknown time-varying arrival rate $\lambda(t)$, with $\rho(t) = \lambda(t)\mathbf{E}[S]$.

The previous policies that we have considered, NEVEROFF, SLEEP and INSTANTOFF, do not satisfy our goal. NEVEROFF and SLEEP are based on a fixed number of servers n^*, and thus do not auto-scale to time-varying demand patterns. INSTANTOFF is actually able to scale capacity in the time-varying case, since it can turn on servers when the load increases, and it can turn *off*



(a)

(b)

Figure 3: Verifying the of accuracy Rule of Thumb #3. This figure shows the relative performance of NEVEROFF, INSTANTOFF and SLEEP policies for a multi-server system, as a function of load, ρ, and mean job size, $\mathbf{E}[S]$, based on simulations. Figure (a) shows NEVEROFF vs. INSTANTOFF. The crosses indicate the region of superiority of INSTANTOFF over NEVEROFF. Figure (b) shows NEVEROFF vs. SLEEP. The crosses indicate the region of superiority of SLEEP over NEVEROFF. The numbers associated with each point denote the % improvement of the superior algorithm over the inferior. The dashed line in Figures (a) and (b) indicate the theoretically predicted split between the policies based on Rule of Thumb #3.

servers when there isn't much work in the system. However, when T_{OFF} is high, we will see that INSTANTOFF performs poorly with respect to ERP.

We propose a new traffic-oblivious auto-scaling policy, DELAYEDOFF, which we will show achieves near-optimal ERP.

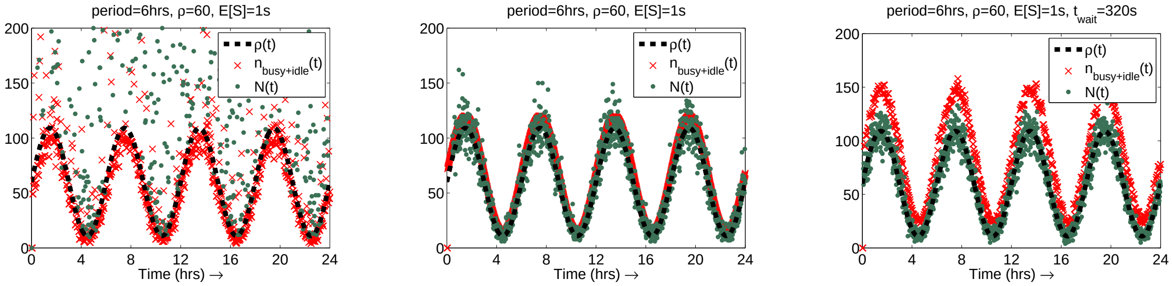
DELAYEDOFF: DELAYEDOFF is a capacity provisioning policy similar to INSTANTOFF, but with two major changes. First, under DELAYEDOFF, we wait for a server to *idle* for some predetermined amount of time, t_{wait}, before turning it *off*. If the server gets a job to service in this period, its idle time is reset to 0. The parameter t_{wait} is a constant chosen independent of load, and thus DELAYEDOFF is a truly traffic-oblivious policy. Second, if an arrival finds more than one servers *idle* on arrival, instead of joining a random *idle* server, it joins the server that was most recently busy (MRB). We will later see that MRB routing is *crucial* to the near-optimality of DELAYEDOFF.

We will demonstrate the superiority of DELAYEDOFF by comparing it against two other policies, the first being INSTANTOFF, and the second being an offline, traffic-aware hypothetical policy, LOOKAHEAD. LOOKAHEAD runs the NEVEROFF policy, with n^* changing as a function of time. LOOKAHEAD smartly calculates $n^*(t)$ for each time t, given the $\rho(t)$ forecast. To do this, we use the idea proposed in [16]. The crux of the idea in [16] is to compute what we will call the "effective load" at time t, $\rho_{\text{eff}}(t)$, as:

$$\rho_{\text{eff}}(t) = \int_{-\infty}^{t} e^{-\mu(t-u)} \lambda(u) du.$$

The quantity $\rho_{\text{eff}}(t)$ denotes the mean number of jobs in the system at time t under the assumption that every job in the system can have its own server. The number of servers to have *on* at time t, $n^*(t)$, is then chosen to be $n^*(t) = \rho_{\text{eff}}(t) + \beta^* \sqrt{\rho_{\text{eff}}(t)}$, where β^* is given by (6).

Figure 4 illustrates the performance of INSTANTOFF, LOOKAHEAD and DELAYEDOFF in the case of a time-varying arrival pattern that resembles a sine curve with a period of 6 hours. In all the simulations, we set $\mathbf{E}[S] = 1sec$, and $T_{OFF} = 200secs$ (hence T_{OFF} is high). Figure 4(a) shows that INSTANTOFF auto-scales poorly as compared to the other policies, in particular $ERP^{INSTANTOFF} \approx 6.8 \times 10^5 Watts \cdot sec$, with $\mathbf{E}[T] \approx 13.17sec$ and $\mathbf{E}[P] \approx 5.19 \times 10^4 Watts$. By contrast, LOOKAHEAD, shown in Figure 4(b), scales very well with the demand pattern. The ERP of LOOKAHEAD is $ERP^{LOOKAHEAD} \approx 1.64 \times 10^4 Watts \cdot sec$, with $\mathbf{E}[T] \approx 1.036sec$ and $\mathbf{E}[P] \approx 1.58 \times 10^4 Watts$. Unfortunately, as pointed out above, LOOKAHEAD requires knowledge of the future arrival pattern to be able to have $n^*(t)$ servers on at time t (in particular, it needs knowledge of the demand curve T_{OFF} units in advance). Thus, while LOOKAHEAD performs very well in a time-varying situation, it is not an online strategy, and is thus, not practical. Figure 4(c) illustrates the excellent auto-scaling capability of DELAYEDOFF for the sinusoidal arrival pattern. Here, $t_{wait} = 320s$ is chosen according to Rule of Thumb #4 presented later in this section. For the case in Figure 4(c), $ERP^{DELAYEDOFF} \approx 1.89 \times 10^4 Watts \cdot sec$ with $\mathbf{E}[T] \approx 1.002sec$ and $\mathbf{E}[P] \approx 1.89 \times 10^4 Watts$. The ERP for DELAYEDOFF is only slightly higher than that of LOOKAHEAD, and far lower than that of INSTANTOFF. DELAYEDOFF slightly overprovisions capacity compared to LOOKAHEAD due to its traffic-oblivious nature. We verify this last observation analytically.



(a) INSTANTOFF

(b) LOOKAHEAD

(c) DELAYEDOFF

Figure 4: Dynamic capacity provisioning capabilities of INSTANTOFF, LOOKAHEAD and DELAYEDOFF. The dashed line denotes the load at time t, $\rho(t)$, the crosses denotes the number of servers that are busy or idle at time t, $n_{busy+idle}(t)$, and the dots represent the number of jobs in the system at time t, $N(t)$.

While analyzing DELAYEDOFF under time-varying traffic is a formidable challenge, we justify its excellent auto-capacity-scaling capabilities in Corollary 1, which shows that under a Poisson arrival process with unknown intensity, DELAYEDOFF achieves near-optimal ERP. Thus, if the rate of change of the arrival rate is less than T_{OFF} (as was the case in Figure 4(c)), we expect DELAYEDOFF to still achieve near-optimal ERP. This is because we are able to turn servers on before the queue builds up.

Theorem 4 *Consider a server farm with Poisson arrival process and Exponential job size distribution. Let ρ denote the average load. Under* DELAYEDOFF *with MRB routing and any constant t_{wait}, with probability $1 - o(1)$, the number of servers on is given by $\rho + \sqrt{\rho \log \rho} + o(\sqrt{\rho \log \rho})$, as $\rho \to \infty$.*

Corollary 1 *If $T_{OFF} = 0$, then DELAYEDOFF achieves optimal ERP asymptotically as $\rho \to \infty$. Specifically, the $ERP^{DELAYEDOFF} \to (\rho * P_{ON} * \mathbf{E}[S])^{-1}$ as $\rho \to \infty$.*

Proof of Corollary 1: From Theorem 4, we know that asymptotically with probability 1, we'll end up with $\rho + \sqrt{\rho \log \rho} + o(\sqrt{\rho \log \rho})$ number of servers on. As mentioned in the proof of Lemma 7, the mean response time for DELAYEDOFF will approach $\mathbf{E}[S]$ as $\rho \to \infty$, since it keeps $\rho + \omega(\sqrt{\rho})$ servers on. Further, the ratio of power consumed by DELAYEDOFF to the minimum power needed to serve jobs $(\rho * P_{ON})$, is $1 + \sqrt{\frac{\log \rho}{\rho}}$, which approaches 1, as $\rho \to \infty$. Thus, the ERP of DELAYEDOFF, with any non-zero t_{wait}, approaches the theoretical lower bound of $(\rho * P_{ON} * \mathbf{E}[S])^{-1}$ as $\rho \to \infty$. ■

Proof of Theorem 4: We first provide an alternate way of viewing the MRB routing. Consider a server farm with infinitely many servers, where we assign a unique rank to each server. Whenever there are n jobs in the server farm, they instantaneously move to servers ranked 1 to n. We now claim that there are m servers on at time t under MRB routing and DELAYEDOFF if and only if there are m servers on at time t in the alternate model under DELAYEDOFF. To see this, let the rank of servers at time t under MRB be defined by the last time they were *idle* (rank 1 server has been idle the shortest and so on). Once a server goes *idle* and gets rank n (thus the number of jobs in the system drops to $n - 1$), its rank remains n until the number of jobs in the system increases to n.

Define the idle period for server $n + 1$, $I(n)$, to be the time that elapses between the instant that the number of jobs in the system transitions from $n + 1$ to n until it next reaches $n + 1$. It is easy to see that the setup delay, T_{OFF} does not affect the distribution of $I(n)$. A rank $n + 1$ server turns *off* when $I(n) > t_{wait}$. The next lemma implies that for any constant $\epsilon > 0$, the mean idle period of $\rho + (1 + \epsilon)\sqrt{\rho \log \rho}$ ranked server goes to ∞, and that of the $\rho + (1 - \epsilon)\sqrt{\rho \log \rho}$ ranked server goes to 0. Due to lack of space, we defer the proof of Lemma 8 to Appendix D.

Lemma 8 *Consider an $M/M/\infty$ system with load ρ. Then, for any constant $\epsilon > 0$:*

$$\lim_{\rho\to\infty} \mathbf{E}\Big[I(\rho + (1+\epsilon)\sqrt{\rho\log\rho})\Big] = \infty$$

$$\lim_{\rho\to\infty} \mathbf{E}\Big[I(\rho + (1-\epsilon)\sqrt{\rho\log\rho})\Big] = 0$$

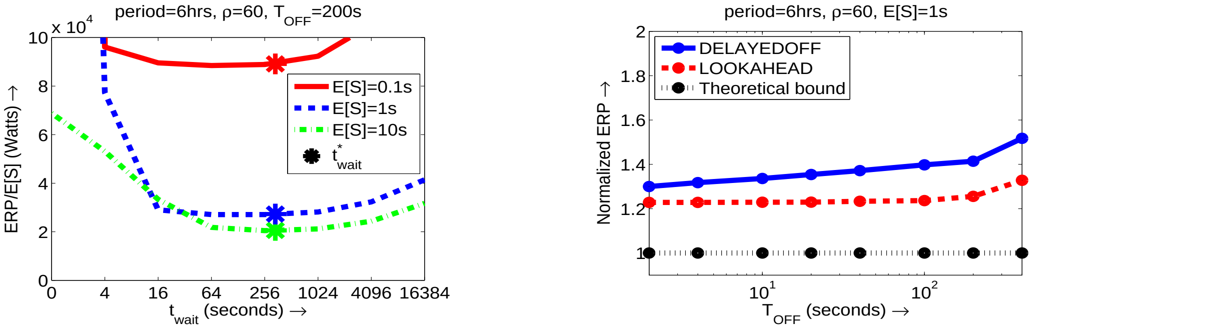
Further, for any constant $\beta > 0$: $\lim_{\rho\to\infty} \sqrt{\rho}\mathbf{E}\Big[I(\rho + \beta\sqrt{\rho})\Big] = \sqrt{2\pi}e^{\beta^2}\Phi(B)$.

Therefore, clearly, for any $\epsilon > 0$, the idle period of server $\rho + (1-\epsilon)\sqrt{\rho\log\rho}$ converges in distribution to 0, and this server is on with probability $1 - o(1)$. It is also easy to show that the mean busy period of server $n = \rho + \delta\sqrt{\rho\log\rho}$ for any $\delta > 0$ is $\mathbf{E}[B(n)] = \frac{1}{\lambda} + o\left(\frac{1}{\lambda}\right) \to 0$. Thus the probability that for any $\epsilon > 0$, the server $n = \rho + (1+\epsilon)\sqrt{\rho}$ is on is upper bounded by $\frac{t_{wait} + \mathbf{E}[B(n)]}{\mathbf{E}[I(n)] + t_{wait} + \mathbf{E}[B(n)]} \to 0$. ■

A natural question that arises is how to choose the optimal value of t_{wait}, which we denote as t^*_{wait}.

Rule of Thumb #4: Choosing t^*_{wait}.

A good choice for the t_{wait} parameter for DELAYEDOFF is $t^*_{wait} \approx T_{OFF} \cdot \frac{P_{ON}}{P_{IDLE}}$. The rule of thumb is along similar lines as the power down strategy proposed in [15] and is based on an amortization argument. Once the server has wasted $P_{IDLE} \cdot t^*_{wait}$ units of power in *idle*, it amortizes the cost of turning the server on later and paying the penalty of $P_{ON} \cdot T_{OFF}$. [3]



(a) Effect of $\mathbf{E}[S]$ on optimal t_{wait}

(b) Effect of T_{OFF} on performance of DELAYEDOFF

Figure 5: (a) Verifying the accuracy of Rule of Thumb #4. The graph shows the effect of t_{wait} on ERP for the DELAYEDOFF policy, in the case of a sinusoidal demand curve, with average $\rho = 60$ and $\mathbf{E}[S] = 0.1, 1, 10s$. Different values of t_{wait} result in different ERP values. However, $t^*_{wait} = T_{OFF} \cdot \frac{P_{ON}}{P_{IDLE}} = 320s$ does well for all values of $\mathbf{E}[S]$. (b) The graph shows the difference in ERP of the DELAYEDOFF and LOOKAHEAD policies. The ERP values are normalized by the theoretical lower bound.

[3] While a reader familiar with work on powering down scheme might find our DELAYEDOFF policy not novel, we would like to point out a conceptual difference between the use of DELAYEDOFF in our work and in the prior literature. The prior literature uses DELAYEDOFF type schemes for stand-alone devices, obtaining constant factor sub-optimality. However, we are applying DELAYEDOFF to each device in a server farm, and are artificially creating an arrival process via MRB so as to make the idle periods of the servers highly variable. This allows DELAYEDOFF to perform near-optimally as ρ increases, that is, the competitive ratio approaches 1. This is not necessarily true under alternate routing schemes, such as probabilistic routing, which would yield a competitive ratio bounded away from 1.

Figure 5(a) verifies Rule of Thumb #4, for different $\mathbf{E}[S]$ values. Figure 5(b) compares the ERP of DELAYEDOFF against the ERP of LOOKAHEAD for different T_{OFF} values. We normalize the ERP values with the theoretical upper bound of $\rho P_{ON} \cdot \mathbf{E}[S]$. Throughout the range of T_{OFF} values, we see that DELAYEDOFF, with t_{wait} chosen based on Rule of Thumb #4, performs within 10% of LOOKAHEAD, based on the ERP. The ERP of both, DELAYEDOFF and LOOKAHEAD are within 70-80% of the ERP values of the theoretical lower bound.

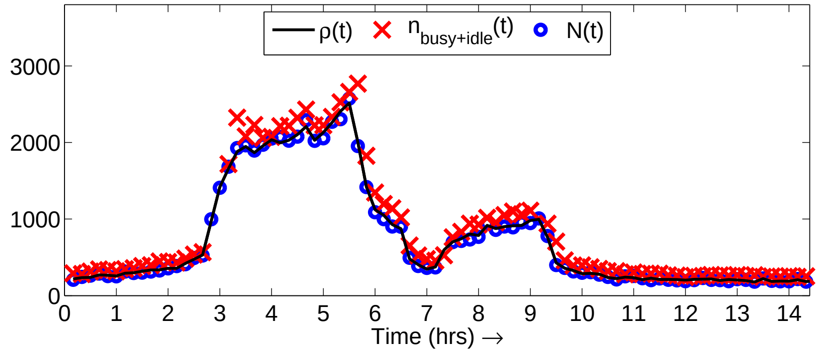


Figure 6: DELAYEDOFF simulation results based on a subset of arrival traces collected from the Internet Traffic Archives, representing 15 hours of bursty traffic during the 1998 Soccer world cup finals. Observe that DELAYEDOFF scales very well even in the case of bursty traffic.

Trace-based simulation results: Thus far we have only looked at simulation results for arrival patterns that look like a sinusoidal curve. However, not all demand patterns are sinusoidal. We now consider a real-life demand pattern based on traces from the 1998 World Cup Soccer website, obtained from the Internet Traffic Archives [1]. The trace contains approximately 90 days worth of arrival data, with more than 1.3 billion arrivals. The data contains very bursty arrivals, with the arrival rate varying by almost a factor of 10, between periods of peak demand and low demand. In particular, the rate of change of arrival rate is sometimes much higher than $T_{OFF} = 200s$. We run DELAYEDOFF on this trace, and compare our results against LOOKAHEAD. Throughout, we assume Exponentially distributed job sizes, with mean 1 second.

Figure 6 shows our simulation results for a subset of the arrival traces, corresponding to the most bursty traffic. We see that DELAYEDOFF (with optimally chosen $t_{wait} = 320s$) adapts extremely well to the time-varying traffic. In fact, over the entire duration of 90 days, the ERP of DELAYEDOFF was within 15% of the ERP of LOOKAHEAD. Thus, we conclude that DELAYEDOFF performs very well even in the case of unpredictable and bursty traffic.

7 Conclusions

This paper address the issue of energy-performance tradeoff in server farms. We utilize the metric of Energy-Response Time Product (ERP) to capture the aforementioned tradeoff. Finding optimal policies to minimize ERP in server farms is an almost intractable problem due to the high dimensionality of the search space of policies, made worse by the numerous *sleep* states present in today's servers. Via the first analysis of the ERP metric, we prove that a very small natural class of server farm management policies suffices to find the optimal or near-optimal policy. We furthermore develop rules of thumb for choosing the best among these policies given the workload and server farm specifications. The impact of our results is two-fold: (i) Our results eliminate the complexity of finding an efficient server farm management policy, and (ii) Our analytical evaluation of the policies advocated in this paper with respect to ERP can guide server designers towards developing a smaller set of *sleep* states with the most impact.

We first prove that for a single server under a Poisson arrival process, the optimal policy with respect to ERP is either to always keep the server *on* or *idle* (NEVEROFF), or to always turn a server *off* when *idle* and to turn it back *on* when work arrives (INSTANTOFF), or to always put the server in some *sleep* state when *idle* (SLEEP). Next we show that for a multi-server system under a Poisson arrival process, the multi-server generalizations of NEVEROFF, INSTANTOFF and SLEEP suffice to find a near-optimal policy. Finally we consider the case of a time-varying demand pattern and propose a simple traffic oblivious policy, DELAYEDOFF, which turns servers on when jobs arrive, but waits for a specific amount of time, t_{wait}, before turning them *off*. Through a clever routing policy, DELAYEDOFF is shown to achieve asymptotic optimality as the load becomes large.

In order to prove the optimality results in this paper, we have made some assumptions: (i) The servers are interchangeable (any job can serve on any server), (ii) The server farm is homogeneous, (iii) The job-sizes are Exponentially distributed (although the asymptotic optimality of DELAYEDOFF extends to general job size distributions). If some or all of these assumptions were to be relaxed, then our optimality results might look different. For example, we might consider policies that treat servers based on their specific characteristics, such as P_{ON}, P_{IDLE} or T_{OFF}. Proving optimality results without the above assumptions is beyond the scope of this paper, and we hope to address some of these issues in a future paper.

References

[1] The internet traffic archives: WorldCup98. Available at http://ita.ee.lbl.gov/html/contrib/WorldCup.html.

[2] I.J.B.F Adan and J. van der Wal. Combining make to order and make to stock. *OR Spektrum*, 20:73–81, 1998.

[3] Susanne Albers and Hiroshi Fujiwara. Energy-efficient algorithms for flow time minimization. *ACM Trans. Algorithms*, 3(4):49, 2007.

[4] Nikhil Bansal, Ho-Leung Chan, and Kirk Pruhs. Speed scaling with an arbitrary power function. In *SODA '09: Proceedings of the Nineteenth Annual ACM -SIAM Symposium on Discrete Algorithms*, pages 693–701, Philadelphia, PA, USA, 2009. Society for Industrial and Applied Mathematics.

[5] Nikhil Bansal, Ho-Leung Chan, and Kirk Pruhs. Speed scaling with an arbitrary power function. In *SODA '09: Proceedings of the Nineteenth Annual ACM -SIAM Symposium on Discrete Algorithms*, pages 693–701, Philadelphia, PA, USA, 2009. Society for Industrial and Applied Mathematics.

[6] Nikhil Bansal, Tracy Kimbrel, and Kirk Pruhs. Dynamic speed scaling to manage energy and temperature. In *FOCS '04: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 520–529, Washington, DC, USA, 2004. IEEE Computer Society.

[7] Nikhil Bansal, Tracy Kimbrel, and Kirk Pruhs. Speed scaling to manage energy and temperature. *J. ACM*, 54(1):1–39, 2007.

[8] Luiz André Barroso and Urs Hölzle. The case for energy-proportional computing. *Computer*, 40(12):33–37, 2007.

[9] S. C Borst, A. Mandelbaum, M. I. Reiman, and Mathematisch Centrum. Dimensioning large call centers. *Operations Research*, 52:17–34, 2000.

[10] Lars Eggert and Joseph D. Touch. Idletime scheduling with preemption intervals. *SIGOPS Oper. Syst. Rev.*, 39(5):249–262, 2005.

[11] R. Gonzalez and M. Horowitz. Energy dissipation in general purpose microprocessors. *IEEE Journal of Solid-State Circuits*, 31(9):1277–1284, 1996.

[12] Shlomo Halfin and Ward Whitt. Heavy-traffic limits for queues with many exponential servers. *Operations Research*, 29(3):567–588, 1981.

[13] Intel Corp. Intel Math Kernel Library 10.0 - LINPACK. http://www.intel.com/cd/software/products/asmo-na/eng/266857.htm, 2007.

[14] Sandy Irani and Kirk R. Pruhs. Algorithmic problems in power management. *SIGACT News*, 36(2):63–76, 2005.

[15] Sandy Irani, Sandeep Shukla, and Rajesh Gupta. Algorithms for power savings. *ACM Trans. Algorithms*, 3(4):41, 2007.

[16] Otis B. Jennings, Avishai M, William A. Massey, and Ward Whitt. Server staffing to meet time-varying demand. *Management Science*, 42:1383–1394, 1996.

[17] Philo Juang, Qiang Wu, Li-Shiuan Peh, Margaret Martonosi, and Douglas W. Clark. Coordinated, distributed, formal energy management of chip multiprocessors. In *ISLPED '05: Proceedings of the 2005 international symposium on Low power electronics and design*, pages 127–130, New York, NY, USA, 2005. ACM.

[18] Chang Woo Kang, Soroush Abbaspour, and Massoud Pedram. Buffer sizing for minimum energy-delay product by using an approximating polynomial. In *GLSVLSI '03: Proceedings of the 13th ACM Great Lakes symposium on VLSI*, pages 112–115, New York, NY, USA, 2003. ACM.

[19] J. Kin, Munish Gupta, and W.H. Mangione-Smith. The filter cache: an energy efficient memory structure. *Microarchitecture, IEEE/ACM International Symposium on*, 0:184, 1997.

[20] Kirk Pruhs, Patchrawat Uthaisombut, and Gerhard Woeginger. Getting the best response for your erg. *ACM Trans. Algorithms*, 4(3):1–17, 2008.

[21] Alma Riska, Ningfang Mi, Evgenia Smirni, and Giuliano Casale. Feasibility regions: exploiting tradeoffs between power and performance in disk drives. *SIGMETRICS Perform. Eval. Rev.*, 37(3):43–48, 2009.

[22] Mircea R. Stan and Kevin Skadron. Power-aware computing: Guest editorial. *IEEE Computer*, 36(12):35–38, December 2003.

[23] U.S. Environmental Protection Agency. EPA Report on server and data center energy efficiency. 2007.

[24] Adam Wierman, Lachlan L. H. Andrew, and Ao Tang. Power-aware speed scaling in processor sharing systems. *INFOCOM*, 2009.

[25] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced cpu energy. *Foundations of Computer Science, Annual IEEE Symposium on*, 0:374, 1995.

A Proof of Theorem 1

Proof of Lemma 1: We first note that if the server is in the *on* state and there is work in the system, then the optimal policy never transitions into a *sleep* state. Suppose, by contradiction, an optimal policy π transitioned into a *sleep* state at time t_0 with work in the queue and then later transitioned through some *sleep* state until finally transitioning to the *on* state at time t_1. We could transform this into a policy π' with equivalent power consumption, but lower mean response time by deferring the powering down until all the work present in the system at t_0 has finished (say at t_2), and then transitioning through the same *sleep* states as π, finally transitioning to the *on* state at time $t_2 + (t_1 - t_0)$.

Next, we note that the only instants at which an optimal policy takes actions will be job completions and job arrivals. Here we assume that once a transition to a *sleep* or *on* state has been initiated, it can't be interrupted. The statement is true due to the assumption of a Poisson arrival process, and the preceding claim that no transitions happen during a busy period.

Finally, we will show that once a policy goes into a *sleep* state when the server goes *idle*, the only other state it will transition to next is *on*. To see this, suppose the server went into *sleep* state S_i. Now, the server will not go into *sleep* state S_j for $j > i$ (and hence to a state with lower power) on a job arrival, otherwise it would have been better to transition to S_j when the server first went *idle*. If the server transitions to a *sleep* state S_k for $k < i$ (thus a state with higher power) but not the *on* state, and later transitions to the *on* state, it would instead have been better to transition directly to the *on* (since the transition times are the same by the *Model Assumptions*), finish processing the work and then transition to state S_k instantaneously. ■

Proof of Lemma 2: The proof proceeds via renewal reward theory. We define a renewal cycle for the server as the time from when a server goes *idle* (has zero work), until it next goes *idle* again. Now consider a specific case, where the server goes into *sleep* state S_i on becoming *idle*, and starts transitioning to the *on* state when n_i jobs accumulate. While the server is turning *on*, there can be more arrivals, and the distribution of number of arrivals is given by X which is distributed as a Poisson random variable with mean λT_{S_i}. Thus, after the server turns *on*, it has $n_i + X$ jobs in the queue, and thus the time until the server goes *idle* is distributed as a sum of $n_i + X$ busy periods of an $M/M/1$ system. The mean number of jobs served in this renewal cycle is given by $\frac{n_i+\mathbf{E}[X]}{1-\rho}$. The expectation of the sum of response times of the jobs that arrive during this renewal cycle can be viewed as a sum of:

1. Sum of waiting times of all jobs before the server turns on (shown in (15) *below).*

2. Sum of the times from when the server turns on until the jobs complete (shown in (16) *below). This can be viewed as a sum of completion times of jobs in $n_i + X$ successive busy periods.*

$$n_i\left(\frac{n_i-1}{2\lambda} + T_{S_i}\right) + \mathbf{E}[X]\frac{T_{S_i}}{2} \tag{15}$$

$$+\frac{1}{1-\rho}\cdot\frac{n_i+\mathbf{E}[X]}{\mu-\lambda} + \mathbf{E}\left[\frac{(n_i+X)(n_i+X-1)}{2(\mu-\lambda)}\right] \tag{16}$$

$$=\frac{1}{1-\rho}\left(\frac{n_i+\mathbf{E}[X]}{\mu-\lambda} + \left[n_iT_{S_i} + \frac{n_i(n_i-1)}{2\lambda} + \frac{\lambda T_{S_i}^2}{2}\right]\right)$$

and the mean number of jobs served in this renewal cycle is $\frac{n_i+\mathbf{E}[X]}{1-\rho}$. Now using renewal reward theory, $\mathbf{E}[T] = \frac{\mathbf{E}[\text{total response time per cycle}]}{\mathbf{E}[\text{number of jobs per cycle}]}$, we have the expression for the mean response time.

The proof for $\mathbf{E}[P]$ is analogous by noting $\mathbf{E}[P] = \frac{\mathbf{E}[\text{total energy per cycle}]}{\mathbf{E}[\text{duration per cycle}]}$. The duration of a cycle is composed of three different times:
1. Time spent waiting for n_i jobs to queue up. This is simply $\frac{n_i}{\lambda}$. The power consumed by the server during this time is P_{S_i}.
2. Time to wake up the server. This is T_{S_i}. The power consumed by the server during this time is P_{ON}.
3. $(n_i + X)$ busy periods. The time it takes for the server to go idle again is simply the duration of $n_i + X$ busy periods. The mean of this duration is $\frac{n_i + \lambda T_{S_i}}{\mu - \lambda}$. The power consumed by the server during this time is P_{ON}.

Thus, we have:

$$\begin{aligned}
\mathbf{E}[P] &= \frac{\mathbf{E}[\text{total energy per cycle}]}{\mathbf{E}[\text{duration per cycle}]} \\
&= \frac{\sum_{i=0}^{N} p_i \sum_{j=1}^{\infty} q_{ij} \left[\frac{j}{\lambda} \cdot P_{S_i} + T_{S_i} \cdot P_{ON} + \frac{j+\lambda T_{S_i}}{\mu-\lambda} \cdot P_{ON}\right]}{\sum_{i=0}^{N} p_i \sum_{j=1}^{\infty} q_{ij} \left[\frac{j}{\lambda} + T_{S_i} + \frac{j+\lambda T_{S_i}}{\mu-\lambda}\right]} \\
&= \frac{\sum_{i=0}^{N} p_i \sum_{j=1}^{\infty} q_{ij} \left(j(\rho P_{ON} + (1-\rho) P_{S_i}) + \lambda T_{S_i} P_{ON}\right)}{\sum_{i=0}^{N} p_i \sum_{j=1}^{\infty} q_{ij} \left(j + \lambda T_{S_i}\right)}.
\end{aligned}$$

■

Proof of Lemma 3: To prove that the optimal strategy is pure, we only need to note that the expressions for both the mean response time and average power are of the form

$$\mathbf{E}[T] = \frac{q_1 t_1 + \ldots + q_n t_n}{q_1 m_1 + \ldots + q_n m_n}$$
$$\mathbf{E}[P] = \frac{q_1 u_1 + \ldots + q_n u_n}{q_1 m_1 + \ldots + q_n m_n},$$

where n is the number of pure strategies that the optimal strategy is randomizing over, for some discrete probability distribution $\{q_1 \ldots, q_n\}$. We will show that when $n = 2$, the optimal strategy is pure, and the proof will follow by induction on n. For $n = 2$, we consider $\mathbf{E}[T]$ and $\mathbf{E}[P]$ as a function of q_1 over the extended domain $q_1 \in (-\infty, +\infty)$, and show that there is no local minima of $\mathbf{E}[T] \cdot \mathbf{E}[P]$ in $q_1 \in (0, 1)$. Further, note that both $\mathbf{E}[T]$ and $\mathbf{E}[P]$ are of the form $a + \frac{b}{c+dq_1}$ for some constants a, b, c, d.
Case 1: Both $\mathbf{E}[T]$ and $\mathbf{E}[P]$ (and hence their product) are increasing or decreasing in q_1, except

for a shared discontinuity at $q_1 = \frac{m_2}{m_2 - m_1}$. In this case, trivially, the minimum of $\mathbf{E}[T] \cdot \mathbf{E}[P]$ in the interval $q_1 \in [0,1]$ is either at $q_1 = 0$ or at $q_1 = 1$.

Case 2: One of $\mathbf{E}[T]$ and $\mathbf{E}[P]$ is an increasing function and the other is a decreasing function of q_1 (except for the shared discontinuity at $q_1 = \frac{m_2}{m_2 - m_1}$). In this case, as $q_1 \to \frac{m_2}{m_2 - m_1}$, $\mathbf{E}[T] \cdot \mathbf{E}[P] \to -\infty$. Second, due to the form of $\mathbf{E}[T]$ and $\mathbf{E}[P]$, it is easy to see that their product has at most one local optimum. Finally, we can see that as $q_1 \to \pm\infty$, $\mathbf{E}[T]\mathbf{E}[P] \to \frac{(t_1 - t_2)(m_1 - m_2)}{(u_1 - u_2)^2}$, which is finite. Combining the previous three observations, we conclude that there is no local minima in the interval $q_1 \in (0,1)$. In other words, in the interval $q_1 \in [0,1]$, the minimum is achieved at either $q_1 = 0$, or $q_1 = 1$. The inductive case for n follows by considering only two variables, q_n and q', where q' is a linear combination of $q_1, q_2, \ldots, q_{n-1}$, and applying the inductive assumption. ■

Proof of Lemma 4: We now know that the optimal power down strategy is of the following form: the server goes into a fixed *sleep* state, S_i, on becoming *idle*. It then waits for some deterministic n_i arrivals before transitioning into the *on* state. We will show that under optimality, $n_i = 1$. The basic idea is to minimize the product of (1) and (3). We first show that if $m = \lambda T_{S_i} > 1$, then the policy where the server goes to *idle* state (recall $T_{IDLE} = 0$) has a lower ERP than going into *sleep* state S_i with *any* n_i. Thus $\lambda T_{S_i} < 1$ is a necessary condition for optimality of *sleep* state S_i.

Lemma 9 *When $\lambda T_{S_i} \geq 1$, NEVEROFF has a lower ERP than a policy involving sleep state S_i with any $n_i > 0$.*

Proof: We will prove the above fact by upper bounding P_{IDLE} by P_{ON}, which only makes the ERP of NEVEROFF worse. Under the above assumption, the ERP values for NEVEROFF and $n_i = n$ are given by:

$$\mathbf{E}[T] \cdot \mathbf{E}[P]|_{NEVEROFF} = \frac{P_{ON}}{\mu - \lambda}$$

$$\mathbf{E}[T] \cdot \mathbf{E}[P]|_{n_i = n} = \left[\frac{\frac{n+m}{\mu-\lambda} + \frac{1}{\lambda}\left(nm + \frac{n^2-n}{2} + \frac{m^2}{2}\right)}{(n+m)^2}\right] \cdot (\rho n + m) P_{ON}, \quad \text{where } m = \lambda T_{S_i}$$

Cross-multiplying the terms, we can say that

$$\begin{aligned}
& \mathbf{E}[T] \cdot \mathbf{E}[P]|_{NEVEROFF} < \mathbf{E}[T] \cdot \mathbf{E}[P]|_{n_i = n} \\
\iff \quad & \rho(n+m)^2 - \left[\rho(n+m) + (1-\rho)\left(\frac{(m+n)^2}{2} - \frac{n}{2}\right)\right](\rho n + m) < 0 \\
\iff \quad & \rho^2\left[-n(m+n) + n\left(\frac{(m+n)^2}{2} - \frac{n}{2}\right)\right] + \rho\left[n(m+n) + (m-n)\left(\frac{(m+n)^2}{2} - \frac{n}{2}\right)\right] \\
& -m\left[\frac{(m+n)^2}{2} - \frac{n}{2}\right] < 0 \qquad (17)
\end{aligned}$$

It is easy to check that the LHS of Eq. (17) is negative at $\rho = 0$, and is zero at $\rho = 1$. Since this expression is quadratic, it suffices to show that the derivative of the above at $\rho = 1$ is positive. This would imply that the curve lies below X-axis in the interval $\rho \in [0, 1)$ for $m, n > 1$. The derivative at $\rho = 1$ is given by:

$$\begin{aligned} & -n(m+n) + (m+n)\left[\frac{(m+n)^2}{2} - \frac{n}{2}\right] \\ & = (m+n)\left[\frac{(m+n)^2}{2} - \frac{3n}{2}\right] \end{aligned}$$

For $m, n > 1$, it is easy to see that $(m+n)^2 > 3n$, and hence the derivative at $\rho = 1$ is indeed positive. ■

Next, we show that when $\lambda T_{S_i} < 1$, the optimal value of n_i is in fact $n_i = 1$. We already know that λT_{S_i} is a necessary condition for the optimality of the pure policy involving S_i, and we thus show that in this case the optimal value of $n_i = 1$. Thus, the optimal policy involving S_i must be SLEEP(S_i).

Lemma 10 *When $\lambda T_{S_i} < 1$, $n_i = 1$ is the optimal policy involving sleep state S_i.*

Proof: Since we know from Lemma 3 that the optimal n_i will be at positive integral values, we can create an alternate function for $\mathbf{E}[P]$ and $\mathbf{E}[T]$ that agrees at these integral points and has continuous derivatives. If this curve finds that the optimal value is indeed $n_{ON} = 1$ then we are done. Let $m = \lambda T_{S_i}$. Further, we assume $P_{S_i} = 0$ as a higher P_{S_i} only favors a lower n_i. These functions are given by:

$$\begin{aligned} \mathbf{E}[T] &= \frac{\frac{x+m}{\mu-\lambda} + \frac{1}{\lambda}\left[x \cdot m + \frac{x^2-x}{2} + \frac{m^2}{2}\right]}{x+m} \\ &= \frac{1}{\mu-\lambda} + \frac{x+m}{2\lambda} - \frac{x}{2\lambda(x+m)} \\ &= \frac{1}{\mu-\lambda} - \frac{1}{2\lambda} + \frac{x+m}{2\lambda} + \frac{m}{2\lambda(x+m)} \\ \mathbf{E}[P] &= \rho \cdot P_{ON} + (1-\rho)\frac{\lambda T_{S_i} \cdot P_{ON}}{x + \lambda T_{S_i}} \\ &= \frac{(\rho x + m)P_{ON}}{x+m} \\ &= \rho \cdot P_{ON} + \frac{m(1-\rho)P_{ON}}{x+m} \end{aligned}$$

The product $\mathbf{E}[T] \cdot \mathbf{E}[P]$ can be written as $ax + b + \frac{c}{x+m} + \frac{d}{(x+m)^2}$. Therefore, there are 3 local optimas, and the second derivative changes sign only once. Further, the curve approaches $-\infty$ when $x \to -\infty$, $+\infty$ when $x \to +\infty$, and again $+\infty$ when $x \to -m$. Further, as $x \to -\infty$, the sign of the second derivative is $-sgn(c)$, and as $x \to +\infty$, the sign of the derivative is $sgn(c)$. In either case, since the curve is convex for some interval in $(-\infty, 0]$, $+\infty$ at $x = -\frac{1}{m}$, and the second derivative changes sign only once, proving that the derivative of $\mathbf{E}[T] \cdot \mathbf{E}[P]$ is positive at $x = 1$ suffices to show that there is no local minima for $x > 1$. (This is because in $[0, +\infty)$, the curve is either convex decreasing at $x = 0$ and then switches to concave, or is convex in the entire interval.)

Taking derivative of the log of the product we get:

$$\frac{\partial}{\partial x} \log\left(\mathbf{E}[T]\mathbf{E}[P]\right) = \frac{1 + \frac{(1-\rho)}{\rho}\left[m + \frac{2x-1}{2}\right]}{x + m + \frac{(1-\rho)}{\rho}\left[mx + \frac{x^2-x}{2} + \frac{m^2}{2}\right]} + \frac{\rho}{\rho x + m} - 2\frac{1}{x+m}$$

$$\implies \frac{\partial}{\partial x} \log\left(\mathbf{E}[T]\mathbf{E}[P]\right)\Big|_{x=1} = \frac{1 + \frac{(1-\rho)}{\rho}\left[m + \frac{1}{2}\right]}{1 + m + \frac{(1-\rho)}{\rho}\left[m + \frac{m^2}{2}\right]} + \frac{\rho}{\rho + m} - 2\frac{1}{1+m}$$

Now,

$$\frac{\partial}{\partial x} \log\left(\mathbf{E}[T]\mathbf{E}[P]\right)\Big|_{x=1} > 0 \iff$$

$$\left[\rho + (1-\rho)(m + \frac{1}{2})\right] \cdot \left[(\rho + m)(1+m)\right] - \left[\rho(1+m) + (1-\rho)(m + \frac{m^2}{m})\right] \cdot \left[\rho(1-m) + 2m\right] > 0$$

The last inequality involves a quadratic in ρ on LHS. It is easy to check that when $\rho = 0$ and $m < 1$, the quadratic is positive. Further, when $\rho = 1$, the value of the quadratic polynomial is 0. Thus it suffices to show that the slope of the above quadratic at $\rho = 1$ is negative (when $m < 1$). This would imply that the above inequality is satisfied in the interval $\rho \in [0, 1)$. Indeed, it can be checked that the derivative at $\rho = 1$ is given by $-\frac{m^3}{2} - \frac{1}{2} < 0$. Thus, we have proven that $n_i > 1$ is not optimal for $m < 1$. Thus, $n_i = 1$ is optimal. ■ ■

B Proof of Theorem 2

The core problem in proving Theorem 2 is in coming up with a tight lower bound for $\mathbf{E}[T]\mathbf{E}[P]$ for the optimal policy. We have a trivial lower bound of $\mathbf{E}[T] \geq \mathbf{E}[S]$, and $\mathbf{E}[P] \geq \rho P_{ON}$. However, this is very loose when ρ is small and T_{OFF} is large.

There are a few **key ideas** in the proof. The first is to give the optimal policy additional capability. We do so by allowing the optimal policy to turn a server *on* from *off* instantaneously (zero setup time). Consequently, each server is either *busy*, *idle*, or *off*. However there is still an energy penalty of $P_{ON}T_{OFF}$. Secondly, we use an accounting method where we charge the energy costs to the jobs, rather than to the server. Thus, each job contributes towards the total response time cost and to the total energy cost. Thirdly, we obtain a lower bound by allowing the optimal policy to choose the state it wants an arrival to see independently for each arrival. This allows us to decouple the decisions taken by the optimal policy in different states. We make this last point clearer next.

An arrival that finds the n jobs in the system (excluding itself) could find the system in one of the following states:

1. At least one server is *idle*: Here, the optimal policy would schedule the arrival on the idle server. In this case, we charge the job $\mathbf{E}[S]$ units for mean response time. Further, the server would have been *idle* for some period before the arrival, and we charge the energy spent during this idle period, as well as the energy to serve the arrival, to the energy cost for the job. However, if under the optimal policy, there is an idle server when the number of jobs increases from n to $n+1$, there must have been a server idle when the number of servers last went down from $n+1$ to n. Furthermore, some server must have remained idle from then until the new arrival which caused the number of jobs to go to $n+1$ (and hence there were no jobs in the queue during this period). Thus, this idle period is exactly the idle period of an $M/M/n+1$ with load ρ, denoted by $I(n)$, where the idle period is defined as the time for the number of jobs to increase from n to $n+1$.
2. No server is *idle*, arrival turns *on* an *off* server: Here, we charge the arrival $\mathbf{E}[S]$ units for mean response time, and $P_{ON}\mathbf{E}[S] + T_{OFF}P_{ON}$ for energy.
3. No server is *idle*, arrival waits for a server to become idle: This case is slightly non-trivial to handle. However, we will lower bound the response time of the job by assuming that the arrival found n servers *busy* with the n jobs. Further, until a departure, every arrival turns on a new server and thus increases the capacity of the system. Thus, this lower bound on queueing time can be expressed as the mean time until first departure in an $M/M/\infty$ system starting with n jobs. We denote this by $D(n)$. The energy cost for the job will simply be $P_{ON}\mathbf{E}[S]$.

We will give the optimal strategy the capability to choose which of the above 3 scenarios it wants for an arrival that occurs with n jobs in the system. Since the response time cost of scenario 1 and 2 are the same, only one of them is used, depending on whether $P_{IDLE}\mathbf{E}[I(n)] > P_{ON}T_{OFF}$ or not. Let $P_{waste}(n) = \min\{P_{IDLE}\mathbf{E}[I(n)], P_{ON}T_{OFF}\}$. Let q_n denote the probability that the optimal policy chooses the best of scenarios 1 and 2 for an arrival finding n jobs in the system, and with probability $1 - q_n$ it chooses scenario 3. Since we are interested in obtaining a lower bound, we will further assume that the probability of an arrival finding n jobs in the system, p_n, is given by the pdf of a Poisson random variable with mean ρ, which is indeed a stochastic lower bound on the stationary number of jobs in the system. We thus obtain the following optimization problem:

$$\begin{aligned}
\mathbf{E}\Big[T^{OPT}\Big]\mathbf{E}\Big[P^{OPT}\Big] &\geq \lambda \min_{\{q_n\}} \left(\mathbf{E}[S] + \sum_n p_n(1-q_n)\mathbf{E}[D(n)]\right)\left(P_{ON}\mathbf{E}[S] + \sum_n p_n q_n P_{waste}(n)\right) \\
&\geq \lambda \min_{\{q_n\}} \left(\sum_n p_n \sqrt{(\mathbf{E}[S] + (1-q_n)\mathbf{E}[D(n)])(P_{ON}\mathbf{E}[S] + q_n P_{waste}(n))}\right)^2 \\
&\text{(By Cauchy-Schwarz inequality)} \\
&= \lambda \left(\sum_n p_n \sqrt{\min\left\{P_{ON}\mathbf{E}[S] + P_{waste}(n), P_{ON}(\mathbf{E}[S] + D(n))\right\}}\right)^2
\end{aligned}$$

The last equality was obtained by observing that the minimum occurs at $q_n = 0$ or $q_n = 1$. The rest of the proof is numerical. We have written a program that computes the above lower bound for a given ρ, T_{OFF}, P_{IDLE} and P_{ON} values. We then compare it against the cost of the NEVEROFF with optimal n^*, and against the following upper bound on the cost of INSTANTOFF: $\lambda P_{ON}\left(\mathbf{E}[S] + T_{OFF}\right)^2$. This upper bound is obtained by forcing every job to run on the server that it chooses to setup on arrival. For each value of ρ, we then search for the T_{OFF} value that maximizes the ratio of the cost of the best of NEVEROFF and INSTANTOFF to the above lower bound, and bound the relative performance of the best of NEVEROFF and INSTANTOFF against the theoretical optimal as a function of ρ and the ratio $\frac{P_{IDLE}}{P_{ON}}$. The above comparison shows that the best of NEVEROFF and INSTANTOFF is within 12% of the optimal policy, when $\rho \geq 20$.

C Proof of Theorem 3

The proof for Theorem 3 proceeds along the same lines as for Theorem 2. For Theorem 3, we have $P_{S_i} > 0$, so the optimal policy does not have infinite servers to work with. Lets say the optimal policy works with N servers. We first add a cost of $\frac{NP_{S_i}}{\lambda}$ to the energy cost of all jobs, and get back a system with $P_{ON} \leftarrow P_{ON} - P_{S_i}$ and $P_{IDLE} \leftarrow P_{IDLE} - P_{S_i}$. We now have the following 3 scenarios that an arrival who sees n jobs in the system could encounter:

1. At least one server is *idle*: In this case we must have $n < N$, and the response time is $\mathbf{E}[S]$ and the energy penalty is $(P_{ON} - P_{S_i})\mathbf{E}[S] + (P_{IDLE} - P_{S_i})I(n)$.

2. Arrival finds no *idle* servers and there is a sleeping server: In this case we may turn on a sleeping server and the energy penalty is $(P_{ON} - P_{S_i})\mathbf{E}[S] + P_{ON}T_{S_i}$. However, the new arrival may be jumping ahead of jobs in the queue. There are at least $(n - (N-1))^+$ of them.

3. Arrival finds no *idle* server and the job waits: In this case the response time is given by $\mathbf{E}[S] + D(n)$ where $D(n)$ denotes the time until first departure in an $M/M/N$ starting with n jobs. The energy cost is just $(P_{ON} - P_{S_i})\mathbf{E}[S]$.

As before, only one of scenarios 1 or 2 is used, and we define $P_{waste} = \min\{P_{ON}T_{S_i}, (P_{IDLE} - P_{S_i})I(n)\mathbf{1}_{n<N}\}$. Our optimization problem then is:

$$\min_{\substack{\{p_n\},\{q_n\} \\ \{p_n\} \geq_{st} Poisson(\rho)}} \lambda \left(\mathbf{E}[S] + \sum_{i=0}^{\infty} p_i(1-q_i)\mathbf{E}[D(n)] \right) \cdot \left(\frac{NP_{S_i}}{\lambda} + (P_{ON} - P_{S_i})\mathbf{E}[S] + \sum_{i=0}^{\infty} p_i q_i P_{waste}(i) \right)$$

The problem with using the above approach is the following: consider a *sleep* state with P_{S_i} very close to P_{IDLE} and $T_{S_i} \ll 1$. In this case, the above problem is optimized for $N = \rho + 1$ (that too because we have a lower bound on N) as follows: for every job, we assume there is a sleeping server which we can wake up for negligible power penalty and negligible response time penalty. Thus we have the following gap in the current accounting method: once there are at least N jobs in the system, a new arrival is allowed to jump ahead of someone in the queue - so either we have jobs in queue, or we have an *idle* server which we are not taking into account. We may try to get around this by not charging jobs for response time when they queue up, but instead charge them for the number of jobs they see. However, we need to argue that the job either pays the penalty of turning on a server, or of waiting. However, we can't charge the job for waiting if we are also charging jobs for queue lengths they see.

To get around this problem, we will charge every job $\mathbf{E}[S]$ units for their service time, $\alpha < 1$ times the cost of the queue lengths they see, and $1-\alpha$ times the cost of their waiting time. We can then optimize over α to get a good lower bound. We now show the steps in detail:

1. At least one server is *idle*: In this case we must have $n < N$, and the response time is $\mathbf{E}[S]$ and the energy penalty is $(P_{ON} - P_{S_i})\mathbf{E}[S] + (P_{IDLE} - P_{S_i})I(n)$.
2. Arrival finds no *idle* servers and there is a sleeping server: In this case we may turn on a sleeping server and the energy penalty is $(P_{ON} - P_{S_i})\mathbf{E}[S] + P_{ON}T_{S_i}$. The response time penalty is $\mathbf{E}[S] + \frac{1}{\lambda}\alpha(\max\{0, n-N+1\})$.
3. Arrival finds no *idle* server and the job waits: In this case the response time is given by $\mathbf{E}[S] + (1-\alpha)D(n)$. The energy cost is just $(P_{ON} - P_{S_i})\mathbf{E}[S]$.

Let $q_{n,1}$ be the probability that scenario 1 is used when there are n jobs, and so on.

Our optimization problem then is:

$$\max_{\alpha} \min_{\{q_{n,1},q_{n,2},q_{n,3}\}|\{p_n\}\geq_{st} Poisson(\rho)} \lambda \left(\mathbf{E}[S] + \sum_{i=0}^{\infty} p_i(q_{i,2}\alpha \frac{(i-N+1)^+}{\lambda} + q_{i,3}(1-\alpha)\mathbf{E}[D(i)] \right)$$
$$\cdot \left(\frac{NP_{S_i}}{\lambda} + (P_{ON} - P_{S_i})\mathbf{E}[S] + \sum_{i=0}^{\infty} p_i(q_{i,1}(P_{IDLE} - P_{S_i})\mathbf{E}[I(i)] + q_{i,2}P_{ON}T_{S_i}) \right)$$

We note that the optimal values for the $q_{i,k} \in \{0,1\}$. Applying Cauchy-Schwarz, we reduce this to a term-by-term minimization, and then we maximize over α.

D Proof of Lemma 8

Without loss of generality, we assume $\mathbf{E}[S] = 1$. Thus $\rho = \lambda$. We begin by writing the recurrences for solving $\mathbf{E}[I(n)]$:

$$\begin{aligned}\mathbf{E}[I(0)] &= \frac{1}{\rho}\\ \mathbf{E}[I(i)] &= \frac{1}{\rho+i} + \frac{i}{\rho+i}\left(\mathbf{E}[I(i-1)] + \mathbf{E}[I(i)]\right)\end{aligned}$$

or equivalently,

$$\begin{aligned}\mathbf{E}[I(i)] &= \frac{1}{\rho} + \frac{i}{\rho}\cdot\mathbf{E}[I(i-1)] = \frac{1}{\rho} + \frac{i}{\rho^2} + \frac{i(i-1)}{\rho^3} + \frac{i(i-1)(i-2)}{\rho^4} + \cdots + \frac{i!}{\rho^{i+1}}\\ &= \frac{i!}{\rho^{i+1}}\left(1 + \frac{\rho}{1} + \frac{\rho^2}{2!} + \cdots + \frac{\rho^{i-2}}{(i-2)!} + \frac{\rho^{i-1}}{(i-1)!} + \frac{\rho^i}{i!}\right)\end{aligned}$$

Now consider $i = \rho + \beta\sqrt{\rho}$. We get:

$$\begin{aligned}\mathbf{E}[I(i)] &= \frac{(\rho+\beta\sqrt{\rho})!e^{\rho}}{\rho^{i+1}}\left(\sum_{k=0}^{\rho+\beta\sqrt{\rho}} e^{-\rho}\frac{\rho^k}{k!}\right) \approx \frac{(\rho+\beta\cdot\sqrt{\rho})!e^{\rho}}{\rho^{i+1}}\Phi(\beta)\\ &\approx \sqrt{\frac{2\pi}{\rho}}\left(1+\frac{\beta}{\sqrt{\rho}}\right)^{\rho+\beta\sqrt{\rho}} e^{-\beta\sqrt{\rho}}\Phi(\beta) \;= \sqrt{\frac{2\pi}{\rho}}\left(1+\frac{\beta}{\sqrt{\rho}}\right)^{\beta\sqrt{\rho}} e^{\rho\log\left[1+\frac{\beta}{\sqrt{\rho}}\right]} e^{-\beta\sqrt{\rho}}\Phi(\beta)\\ &\approx \sqrt{\frac{2\pi}{\rho}}e^{\beta^2}e^{\rho(\frac{\beta}{\sqrt{\rho}}-\frac{\beta^2}{2\rho}+o(1/\rho))}e^{-\beta\sqrt{\rho}}\Phi(\beta)\\ &\approx \frac{\sqrt{2\pi}e^{\beta^2}\Phi(B)}{\sqrt{\rho}}\end{aligned}$$

which proves the second part of the theorem.

Now consider $i = \rho + \eta\sqrt{\rho \log \rho}$ for some constant $\eta > 0$:

$$\begin{aligned}
\mathbf{E}[I(i)] &\approx \frac{(\rho + \eta\sqrt{\rho\log\rho})!e^{\rho}}{\rho^{\rho+\eta\sqrt{\rho\log\rho}+1}} \;\approx \sqrt{\frac{2\pi}{\rho}}\left(1 + \frac{\eta\sqrt{\rho\log\rho}}{\rho}\right)^{\rho+\eta\sqrt{\rho\log\rho}} e^{-\eta\sqrt{\rho\log\rho}} \\
&= \sqrt{\frac{2\pi}{\rho}} e^{(\rho+\eta\sqrt{\rho\log\rho})\log\left(1+\frac{\eta\sqrt{\rho\log\rho}}{\rho}\right)} e^{-\eta\sqrt{\rho\log\rho}} \\
&\approx \sqrt{\frac{2\pi}{\rho}} e^{(\rho+\eta\sqrt{\rho\log\rho})\left(\frac{\eta\sqrt{\rho\log\rho}}{\rho} - \frac{\eta^2\rho\log\rho}{2\rho^2} + \theta\left(\frac{(\eta\sqrt{\rho\log\rho})^3}{\rho^3}\right)\right) - \eta\sqrt{\rho\log\rho}} \\
&\approx \sqrt{\frac{2\pi}{\rho}} e^{\frac{\eta^2\rho\log\rho}{2\rho}} \;= \sqrt{2\pi}\rho^{\frac{\eta^2-1}{2}}
\end{aligned}$$

Thus for $\eta^2 > 1$ $\mathbf{E}\Big[I(\rho + \eta\sqrt{\rho\log\rho})\Big] \to \infty$, and for $\eta^2 < 1$ $\mathbf{E}\Big[I(\rho + \eta\sqrt{\rho\log\rho})\Big] \to 0$ as $\rho \to \infty$.