Dynamic Control of Running Servers [⋆]

Esa Hyytiä[1,2], Douglas Down[3], Pasi Lassila[2], and Samuli Aalto[2]

[1] Department of Computer Science, University of Iceland, Iceland
[2] Department of Communications and Networking, Aalto University, Finland
[3] Department of Computing and Software, McMaster University, Canada

Abstract. Motivated by a data center setting, we study the problem of joint dispatching and server sleep state control in a system consisting of two queues in parallel. Using the theory of Markov decision processes and a novel lookahead approach, we explicitly determine near-optimal control policies that minimize a combination of QoE costs, energy costs, and wear and tear costs due to switching. Guidelines are provided as to when these combined policies are most effective.
Keywords: data centers; queueing systems; dynamic control; Markov decision processes; lookahead techniques

1 Introduction

Server clusters comprise the core of modern data centers and cloud computing systems. Stochastic queueing models, such as multiserver systems with a central queue or distributed systems consisting of multiple parallel servers with their own queues, are suitable for the performance analysis of such systems [11]. Traditional mechanisms for their control include job scheduling and dispatching (a.k.a. task assignment). In a distributed system, the dispatcher decides to which server an arriving job is routed, and the local scheduler decides on how the service capacity of the server is dynamically shared among its jobs.

When optimizing the control of such queueing systems, an important measure is the response time, i.e., the total delay of a job. Typical objectives related to the delay performance are minimization of the mean response time or its tail probability. However, in current computing systems, a significant additional factor is energy efficiency [1]. It is not enough to optimize the delay performance, but one should also take into account the energy aspect. Both dispatching and scheduling decisions affect not only the delay performance but also the energy efficiency. An additional dimension in this joint control problem is related to the sleep states of servers. One should decide on when to put a server into a sleep mode and when to wake it up again. While sleeping tends to reduces energy costs, it has a negative effect on the delay performance, since, after the wake-up, there is typically a relatively long setup time before the server is back in full operation. In addition, the more often a server is switched on and off, the more vulnerable to failures it becomes.

[⋆] This work was supported by the Academy of Finland in the FQ4BD and TOP-Energy projects (grant nos. 296206 and 268992).

In this paper, we consider distributed systems consisting of multiple parallel servers with their own queues. The servers are assumed to apply the First Come First Served (FCFS) scheduling policy. Our goal is to develop near-optimal joint control policies for dispatching and sleep state control of servers when the trade-off between delay performance and energy efficiency is described by a composite objective function. Our approach is based on the theory of Markov decision processes (MDPs) [25]. More precisely said, we apply the *policy improvement* method and combine it to a *lookahead* technique [13]. We demonstrate that the combination of these control mechanisms can yield significant savings, in particular when the system is under moderate load.

2 Related Work

While queueing theory has been applied for decades to evaluate the performance of computing systems, Chen et al. [2] and Sledgers et al. [26] were the first to use queueing models to study the problem of energy-aware control of server clusters. Since then, energy-aware multiserver systems with a central queue have been analyzed in many papers [6, 5, 19, 4, 20, 23, 18, 24]. However, the optimal control problem in such energy-aware multiserver systems has proved difficult. Exact solutions have been found for the single server case [5, 16, 8–10], but with multiple servers only structural properties of the optimal policy are obtained [17].

For static dispatching policies (such as Random Routing), the analysis of energy-aware distributed systems consisting of multiple parallel servers with their own queues is straightforward, since the parallel queues can be analyzed separately. However, dynamic policies (such as Join the Shortest Queue) are mathematically tractable only under restrictive assumptions. Moreover, exact optimality results are scarce, being available only for some specific setups. Near-optimal solutions for dispatching problems have been developed by applying the policy iteration approach from the theory of Markov decision processes [25, 28, 12]. Such an approach has been utilized for composite objective functions that take into account the performance-energy trade-off [22, 14, 15].

With respect to sleep state control in a multiserver setting, it has been observed in [5, 7] that putting servers to sleep aggressively can be harmful. If a server is turned off e.g., immediately when it becomes idle, energy costs may be saved in the short term, but significant response time degradation may result, in particular when setup times are significant. They suggest that an idle server waits a period of time before being put to sleep. They choose a state-independent timer for this wait and design a dispatching policy that takes this sleep state control mechanism into account. In [21], it is shown that dispatching and sleep state control of this form is optimal (simultaneously minimizes mean response time and energy costs) in a many-servers asymptotic regime. In contrast, we allow the sleep state control to be state dependent, considering the joint control problem in an MDP framework. Consistent with [5, 7, 21], we see that gains can be made by conscious turn off decisions, but for the small system that we consider, we further identify system parameters under which such gains are significant.

3 Joint Control Problem

Consider a distributed computing system consisting of two parallel FCFS servers with their own queues. The servers are assumed to be homogeneous, i.e., they have the same service rate. Service times of jobs, denoted by X, are assumed to be independent and generally distributed with finite first two moments. Jobs arrive according to a Poisson process with rate Λ. The load per server is denoted by $\rho = \Lambda\,\mathbb{E}[X]/2$. For stability, we assume that $\rho < 1$. Each job is dispatched to one of the two servers upon its arrival. The dispatching decisions are assumed to be *dynamically controllable*, i.e., they may depend on the state of the system.

The servers are assumed to be *energy-aware*, and there are four different *operational states* for the servers: (i) *busy*, (ii) *idle*, (iii) *off*, and (iv) *set up*. A server is busy when it is processing jobs. As soon as the service of all available jobs is completed, the server becomes idle. The server remains idle as long as one of the following events takes place. Either a new job is dispatched to it, in which case the server becomes again busy and starts serving the new job, or the *switch-off timer* (associated with the idle server) expires, in which case the server is immediately switched off. The length of the switch-off timer, denoted by τ, is assumed to be dynamically controllable. If the switch-off timer expires, the server remains off until a new job is dispatched to it, at which time the server is switched on (set up). After a *setup time*, the server becomes again busy and starts serving the jobs waiting in its queue. Setup times of servers, denoted by D, are assumed to be independent and generally distributed with finite first two moments. In particular, we are, however, interested in the case where the setup times are deterministic, $D = d$. When busy or set up, the power consumption of a server is e [watts], but when idle, its power consumption is ϵ [watts], which is assumed to be less than e, i.e., $\epsilon = \gamma e$, where $\gamma < 1$. In the sequel, we will use e as our power unit. When off, a server does not consume any power. In line with [5], such an energy-aware server is called *DelayedOff*. Special cases are *NeverOff* ($\tau \to \infty$) and *InstantOff* ($\tau = 0$).

The cost structure comprises both QoE and system specific cost components including both energy and switching costs. The QoE metric in our model is the mean response time $\mathbb{E}[T]$. Note that, due to the well-known Little's formula, minimizing the mean response time is equivalent to minimizing the mean total number of jobs in the system, $\mathbb{E}[N] = \Lambda\,\mathbb{E}[T]$. Energy costs are related to the mean total power consumption $\mathbb{E}[P]$, and switching costs take into account wear and tear costs of switching a server off and on. More precisely, we assume that the *mean cost rate* of the whole system is given by

$$r = r_T + r_P + r_S = \mathbb{E}[N] \cdot c_T + \mathbb{E}[P] \cdot c_P + \Lambda_S \cdot c_S, \tag{1}$$

where Λ_S denotes the aggregate switch-on (and off) rate of the two servers. The constants (c_T, c_P, c_S) map each component to a common unit.

Now the problem is to find a *joint dispatching and sleep state control* that minimizes the mean cost rate (1). Dispatching decisions are made when new jobs arrive, and the sleep states are controlled when a server becomes idle or a new job arrives. We allow dynamic control that is based on the current state of the system, together with the service time of the arriving job if we are about to make a dispatching decision. We assume that the state of a server is described by its *virtual backlog*, switch-off timer value, and energy state. The virtual backlog u refers to the time needed to complete the service of all jobs currently in the system (without any new arrivals). If the server is busy, the virtual backlog is just the ordinary backlog, i.e, the sum of remaining service times, but if the server is in set up, it also includes the remaining setup time. For an off or idle server, the virtual backlog equals 0. The current value of the switch-off timer, t, refers to the time that the server has been (continuously) idle, $0 \leq t \leq \tau$. In the following section, we tackle this optimal control problem by the policy improvement method combined with the lookahead technique.

4 Policy Improvement and Lookahead

For the policy improvement method, we need a *basic control policy* that can be analyzed *explicitly.* Such a policy is attained if we apply random routing to dispatching and deterministic switch-off timers for the sleep state control. In this paper, we choose uniform routing probabilities (1/2 for each server) so that the load is balanced, which is a reasonable basic dispatching policy. As a result, there are two independent single-server queues with Poisson arrivals at rate $\lambda = \Lambda/2$. We need to derive (for each queue i) the so-called *relative value function* $v_i(u_i, t_i) - v_i(0, 0)$, which gives the difference in the mean accumulated costs if the system starts from states (u_i, t_i) and $(0, 0)$, respectively, where u_i refers to the virtual backlog and t_i the switch-off timer value of server i. Formally, the *value function* is defined as

$$v_i(u_i, t_i) := \lim_{t \to \infty} \mathbb{E}\left[C_i(u_i, t_i, t) - r_i t\right], \tag{2}$$

where $C_i(u_i, t_i, t)$ denotes the costs queue u incurs during time $(0, t)$ when initially in state (u_i, t_i), and r_i is the mean cost rate of queue i. The relative value function for the DelayedOff M/G/1-FCFS queue is derived in Sect. 5. In addition, we assume that the sleep state control of the basic policy is such that server 1 is an ordinary NeverOff server ($\tau_1 \to \infty$) and server 2 is an energy-aware InstantOff server ($\tau_2 = 0$), which is a reasonable compromise for all traffic load situations.

Below we show how to improve this static (i.e., state-independent) basic policy by developing a dynamic control policy that utilizes the state information. We start from the dispatching decisions. For the sleep state control, we consider separately two different cases: first the case when a server becomes idle, and thereafter the case when a server is already off and a new job arrives. For simplicity, the results in this section are given for deterministic setup times d.

4.1 Improving Dispatching Decisions

Recall first that the basic policy assumes that server 1 is NeverOff ($\tau_1 \to \infty$) and server 2 InstantOff ($\tau_2 = 0$). Thus, the state of server 1 is completely described by the virtual backlog u_1 (which, in this case, is the same as the ordinary backlog). From Prop. 1 (presented and justified in Sect. 5), we get its relative value function:

$$v_1(u_1) - v_1(0) = \frac{\lambda\, u_1^2}{2(1-\rho)}\, c_T + u_1(1-\gamma)c_P. \tag{3}$$

On the other hand, to describe the state of server 2, it is enough to specify the virtual backlog u_2 and indicate whether the server is switched off (s) or running (r), i.e., in set up or busy. From Prop. 1, we again get the corresponding relative value function:

$$v_2^{(r)}(u_2) - v_2^{(s)}(0) = \frac{\lambda\, u_2^2}{2(1-\rho)}\, c_T + \frac{u_2}{1+\lambda d}\left[c_P - \lambda c_S - \frac{\lambda d(2+\lambda d)}{2(1-\rho)}\, c_T\right]. \tag{4}$$

The dispatching decisions of the static basic policy can be improved by choosing the server i for which the expected *admission costs* $a_i(u_i, x)$ are minimized, where x denotes the service time of the arriving job. The expected admission costs can be calculated as follows. For server 1, we have

$$\begin{aligned} a_1(u_1, x) &= (u_1 + x)c_T + v_1(u_1 + x) - v_1(u_1) \\ &= \left(u_1 + x + \frac{\lambda x(2u_1 + x)}{2(1-\rho)}\right) c_T + x(1-\gamma)c_P, \end{aligned}$$

and, for server 2, we have

$$\begin{aligned} a_2^{(r)}(u_2, x) &= (u_2 + x)c_T + v_2^{(r)}(u_2 + x) - v_2^{(r)}(u_2) \\ &= \left(u_2 + x + \frac{\lambda x(2u_2 + x)}{2(1-\rho)}\right) c_T + \frac{x}{1+\lambda d}\left(c_P - \lambda c_S - \frac{\lambda d(2+\lambda d)}{2(1-\rho)} c_T\right), \end{aligned}$$

and $a_2^{(s)}(0, x) = c_S + a_2^{(r)}(0, d + x)$, obviously, as one switching cost is saved. In each case, it is easy to identify the immediate cost consisting of the response time of the new job and the possible switching cost c_S.

4.2 Lookahead for Server Switch-Off

The static decision to switch server 2 off whenever it becomes idle is obviously suboptimal. Next we apply the lookahead technique to tackle this [13].

Suppose that server 1 has backlog u_1 when server 2 becomes idle. By default, we would switch server 2 off at this point. However, we can consider the following two alternative actions:

A: Switch server 2 off immediately and route the next job, given it arrives before time τ, to server 1.

B: Keep server 2 running idle for time τ hoping that a new job arrives soon, which would then be routed to server 2.

In both cases, after time τ, we return back to the default routing and switch-off policies. In general, τ is a free parameter less than u_1.

With these, one can compute *in closed-form* the expected cost of the alternative actions, denoted by d_A and d_B, respectively,

$$\begin{aligned} d_A = &\int_0^\tau \Lambda e^{-\Lambda t}\left[(c_P - r)t + (u_1 - t + \mathbb{E}[X])c_T + \mathbb{E}[v_1(u_1 - t + X)] + v_2^{(s)}(0)\right] dt \\ &+\ e^{-\Lambda\tau}((c_P - r)\tau + v_1(u_1 - \tau) + v_2^{(s)}(0)), \\ d_B = &\int_0^\tau \Lambda e^{-\Lambda t}\left[((1+\gamma)c_P - r)t + \mathbb{E}[X]\ c_T + v_1(u_1 - t) + \mathbb{E}\left[v_2^{(r)}(X)\right]\right] dt \\ &+\ e^{-\Lambda\tau}(((1+\gamma)c_P - r)\tau + v_1(u_1 - \tau) + v_2^{(s)}(0)). \end{aligned}$$

Then we choose to keep server 2 idle if that action yields a smaller expected cost, $d_A - d_B > 0$.

As time passes without an arrival, u_1 gets smaller, and the benefits from keeping server 2 running idle become smaller. This suggests that we can consider a differential time step. In particular, we find that

$$\begin{aligned} f(u_1) &:= \lim_{\tau\to 0} \frac{d_B - d_A}{\tau} \\ &= \frac{\lambda}{1-\rho}\left(\frac{(\rho d(2+\lambda d))}{1+\lambda d} + 2u_1\right) c_T + \frac{2\lambda\rho(d\,c_P + c_S)}{1+\lambda d} - (1+2\rho)\gamma\, c_P, \end{aligned}$$

and then solving $f(u_1) = 0$ yields the critical backlog above which server 2 can be kept idle instead of being switched off,

$$u_1^* = \frac{1+\rho-2\rho^2}{2\lambda c_T}\gamma\, c_P - \frac{\rho(2(1-\rho)(d\,c_P + c_S) + d(2+\lambda d)c_T)}{2c_T(1+\lambda d)}.$$

Note that u_1^* depends only on the first moment of the service time distribution. Moreover, the second term is always negative, and therefore if $\epsilon = 0$, i.e., $\gamma = 0$, then $u_1^* < 0$, which suggests that it is preferable to keep server 2 on, which of course makes sense if idling incurs no energy costs.

Alternatively, one can determine the critical energy cost denoted by c_P^* above which server 2 should be switched off.

$$c_P^* = \frac{\lambda}{\gamma(1+2\rho)(1+\lambda d) - 2\rho\lambda d}\left(2\rho c_S + \frac{2u_1(1+\lambda d) + d\rho(2+\lambda d)}{1-\rho}c_T\right),$$

and in the special case of $\epsilon = e$, i.e., $\gamma = 1$, we have

$$c_P^* = \frac{\lambda}{1+\lambda d + 2\rho}\left(2\rho c_S + \frac{2u_1(1+\lambda d) + d\rho(2+\lambda d)}{1-\rho}c_T\right).$$

Numerical Example Let us next assume unit service time, $\mathbb{E}[X] = 1$, unit response time cost, $c_T = 1$, unit setup delay, $d = 1$, and no switching costs, $c_S = 0$.

Moreover, the energy cost in busy/setup states is $c_P = 1$, and in the idle state γc_P, where $\gamma \in \{0.5, 1\}$. Server 1 has backlog u_1, and server 2 becomes empty. Then we vary the offered load ρ and evaluate when one should keep server 2 running. The results are depicted in Figure 1(a). We can see that as the load increases, the critical backlog decreases eventually becoming zero, i.e., if the system is heavily loaded, then the response time costs starts to dominate (cf. the knee in the response time curve).

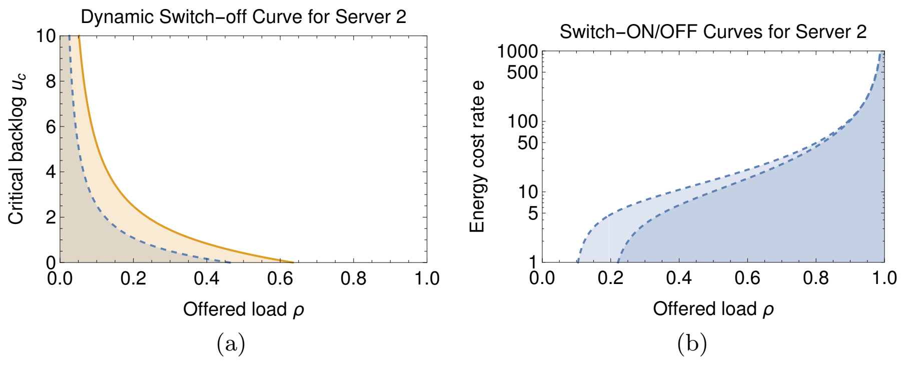

Fig. 1. Dynamic switch-off policies when $c_P = 1$, $c_T = 1$, $c_S = 0$ and $\gamma \in \{0.5, 1\}$ (left). Dynamic switch-on policy resulting from the lookahead analysis (right).

4.3 Proactive Switch-On of Servers

A similar lookahead analysis can be performed for a system in state $(u_1, 0)$, where server 2 has been switched off. As u increases, at some point it may be beneficial to switch server 2 back on, as the next job most likely ends up there. Perhaps the most elementary lookahead action to consider in this case is the action that switches server 2 on and routes the next job there unconditionally. This is a simple decision, and, e.g., server 1 may empty meanwhile and we keep then both servers running idle until the next job arrives. Carrying out a similar analysis and solving for the critical energy cost rate, we get

$$c_P^* = \frac{\lambda\left(e^{-2\lambda d} - (2-\rho)\right)}{(\gamma - (1-\gamma)\lambda d)\left(e^{-2\lambda d} + \rho\right) + 2\lambda d} c_S + \\ \frac{(1+\lambda d)\left(\lambda d\left(e^{-2\lambda d} + \rho\right) + e^{-2\lambda u_1} - 1 + 2\lambda u_1\right) - \lambda d\,(1-\rho) + 1 - e^{-2\lambda d}}{2(1-\rho)\left((\gamma - (1-\gamma)\lambda d)\left(e^{-2\lambda d} + \rho\right) + 2\lambda d\right)} c_T .$$

Numerical Example Figure 1(b) shows the keep running and proactive switching on decisions for with $\mathbb{E}[X] = 1$, $\epsilon = 1$, $c_S = 0$, $d = 1$, $c_T = 1$ and $u_1 = 5$.

5 Value function for an energy-aware single server queue

In this section, we consider a generic DelayedOff M/G/1-FCFS queue with arrival rate λ, generally distributed service times X, deterministic switch-off timer

τ, and generally distributed setup times D. We assume a stable system, i.e., $\rho = \lambda \mathbb{E}[X] < 1$.

Our purpose is to derive the relative value function $v(u,t) - v(0,0)$, where u refers to the current virtual backlog and t the current value of the switch-off timer, $0 \le t \le \tau$. Thus, either u, t, or both are zero. The reference state is the renewal point when the server becomes idle and the switch-off timer starts to count towards τ.

We start by giving the mean number of jobs $\mathbb{E}[N]$, the mean power consumption $\mathbb{E}[P]$, and the mean switch-off rate λ_S for this DelayedOff M/G/1-FCFS queue, which are derived, e.g., in [8]:

$$\mathbb{E}[N] = \rho + \frac{\lambda^2 \mathbb{E}\left[X^2\right]}{2(1-\rho)} + \frac{\lambda(2\,\mathbb{E}[D] + \lambda\,\mathbb{E}\left[D^2\right])}{2\left(\lambda\,\mathbb{E}[D] + e^{\lambda\tau}\right)}, \tag{5}$$

$$\mathbb{E}[P] = \frac{\left(\lambda\,\mathbb{E}[D] + \rho e^{\lambda\tau}\right) + \gamma(1-\rho)\left(e^{\lambda\tau} - 1\right)}{\lambda\,\mathbb{E}[D] + e^{\lambda\tau}}, \tag{6}$$

$$\lambda_S = \frac{\lambda(1-\rho)}{\lambda\,\mathbb{E}[D] + e^{\lambda\tau}}. \tag{7}$$

Similarly as for the whole system in (1), the mean cost rate for the single server queue consists of three terms,

$$r = r_T + r_P + r_S = \mathbb{E}[N] \cdot c_T + \mathbb{E}[P] \cdot c_P + \lambda_S \cdot c_S. \tag{8}$$

The value function $v(u,t)$ is also a composite function consisting of three corresponding terms,

$$v(u,t) = v_T(u,t) + v_P(u,t) + v_S(u,t). \tag{9}$$

Proposition 1. *For a DelayedOff M/G/1-FCFS queue, the components of the relative value function $v(u,t) - v(0,0)$ are as follows:*

$$v_T(u,t) - v_T(0,0) = \frac{1}{2(1-\rho)}\left(\lambda u^2 - \frac{\left(2\,\mathbb{E}[D] + \lambda\,\mathbb{E}\left[D^2\right]\right)\left(\lambda u + 1 - e^{\lambda t}\right)}{\lambda\,\mathbb{E}[D] + e^{\lambda\tau}}\right) c_T,$$

$$v_P(u,t) - v_P(0,0) = \frac{\left((1-\gamma)e^{\lambda\tau} + \gamma\right)\lambda u - \left(\gamma - \lambda\,\mathbb{E}[D]\,(1-\gamma)\right)\left(e^{\lambda t} - 1\right)}{\lambda\left(\lambda\,\mathbb{E}[D] + e^{\lambda\tau}\right)} c_P,$$

$$v_S(u,t) - v_S(0,0) = -\frac{\lambda u + 1 - e^{\lambda t}}{\lambda\,\mathbb{E}[D] + e^{\lambda\tau}} c_S.$$

Proof. 1° Let us start with the response time related costs. By (5) and (8), we get

$$r_T = \mathbb{E}[N] \cdot c_T = \left(\rho + \frac{\lambda^2 \mathbb{E}\left[X^2\right]}{2(1-\rho)} + \frac{\lambda(2\,\mathbb{E}[D] + \lambda\,\mathbb{E}\left[D^2\right])}{2\left(\lambda\,\mathbb{E}[D] + e^{\lambda\tau}\right)}\right) c_T. \tag{10}$$

1.1° Assume first $u > 0$ and $t = 0$ so that the server is busy or in set up. Let B_u denote the length of the resulting "busy period", i.e., the time needed to decrease the virtual backlog from u to 0. In addition, let N_u denote the total number of jobs that arrived during that time, and $\mathbb{E}[T_1 + \ldots + T_{N_u}]$ the sum of their expected response times. By considering a separate M/G/1 queue with arrival rate λ where the service time of the first customer of each busy period equals u but for the other customers the service time follows the distribution of X, we get easily (see, e.g., [27])

$$\begin{aligned} \mathbb{E}[B_u] &= \frac{u}{1-\rho}, \qquad (11) \\ \mathbb{E}[T_1 + \ldots + T_{N_u}] &= \frac{1}{2(1-\rho)}\left(\lambda u^2 + 2\rho u + \frac{\lambda^2\,\mathbb{E}\left[X^2\right]u}{2(1-\rho)}\right). \end{aligned}$$

Now, for the value function at state $(u, 0)$, from (2), we have

$$v_T(u,0) = \mathbb{E}[T_1 + \ldots + T_{N_u}]\,c_T - \mathbb{E}[B_u]\,r_T + v_T(0,0),$$

which implies, by (10) and the previous expressions, that

$$v_T(u,0) - v_T(0,0) = \frac{1}{2(1-\rho)}\left(\lambda u^2 - \frac{(2\,\mathbb{E}[D] + \lambda\,\mathbb{E}\left[D^2\right])\lambda u}{\lambda\,\mathbb{E}[D] + e^{\lambda\tau}}\right)c_T.$$

1.2° Assume now that $u = 0$ and $0 \le t \le \tau$ so that the server is idle or off. For the value function at state $(0, 0)$, we clearly have

$$\begin{aligned} v_T(0,0) &= \mathbb{E}\left[\int_0^t \lambda e^{-\lambda s}\left(-s r_T + X c_T + v_T(X,0)\right)ds\right] + e^{-\lambda t}(-t r_T + v_T(0,t)) \\ &= -\frac{1-e^{-\lambda t}}{\lambda}r_T + (1-e^{-\lambda t})(\mathbb{E}[X]\,c_T + \mathbb{E}[v_T(X,0)]) + e^{-\lambda t}v_T(0,t). \end{aligned}$$

By (10) and the result of 1.1°, we get, after some manipulations,

$$v_T(0,t) - v_T(0,0) = \frac{\left(2\,\mathbb{E}[D] + \lambda\,\mathbb{E}\left[D^2\right]\right)\left(e^{\lambda t} - 1\right)}{2(1-\rho)(\lambda\,\mathbb{E}[D] + e^{\lambda\tau})}c_T.$$

2° Let us now consider the energy related costs. By (6) and (8), we get

$$r_P = \mathbb{E}[P]\cdot c_P = \frac{\left(\lambda\,\mathbb{E}[D] + \rho e^{\lambda\tau}\right) + \gamma(1-\rho)\left(e^{\lambda\tau} - 1\right)}{\lambda\,\mathbb{E}[D] + e^{\lambda\tau}}c_P. \qquad (12)$$

2.1° Assume again first that $u > 0$ and $t = 0$. Let B_u denote the same "busy period" as in 1.1° so that (11) holds. For the value function at state $(u, 0)$, we have

$$v_P(u,0) = \mathbb{E}[B_u]\,(c_P - r_P) + v_P(0,0),$$

which implies, by (12) and the previous expression, that

$$v_P(u,0) - v_P(0,0) = \frac{\left((1-\gamma)e^{\lambda\tau} + \gamma\right)u}{\lambda\,\mathbb{E}[D] + e^{\lambda\tau}}\,c_P.$$

2.2° Assume now that $u = 0$ and $0 \leq t \leq \tau$. For the value function at state $(0, 0)$, we clearly have

$$\begin{aligned} v_P(0,0) &= \mathbb{E}\left[\int_0^t \lambda e^{-\lambda s}\left(s(\gamma c_P - r_P) + v_P(X,0)\right) ds\right] + e^{-\lambda t}(t(\gamma c_P - r_P) + v_P(0,t)) \\ &= \frac{1-e^{-\lambda t}}{\lambda}(\gamma c_P - r_P) + (1-e^{-\lambda t})\mathbb{E}\left[v_P(X,0)\right] + e^{-\lambda t} v_P(0,t). \end{aligned}$$

By (12) and the result of 2.1°, we get, after some manipulations,

$$v_P(0,t) - v_P(0,0) = -\frac{\left(\gamma - \lambda\,\mathbb{E}\left[D\right](1-\gamma)\right)\left(e^{\lambda t} - 1\right)}{\lambda\left(\lambda\,\mathbb{E}\left[D\right] + e^{\lambda \tau}\right)}\, c_P.$$

3° Consider finally the switching costs. By (7) and (8), we get

$$r_S = \lambda_S \cdot c_S = \frac{\lambda(1-\rho)}{\lambda\,\mathbb{E}\left[D\right] + e^{\lambda\tau}}\, c_S. \tag{13}$$

3.1° As before, assume first that $u > 0$ and $t = 0$. Let B_u denote the same "busy period" as in 1.1° and 2.1° so that (11) holds. For the value function at state $(u, 0)$, we have

$$v_S(u,0) = -\mathbb{E}\left[B_u\right] r_S + v_S(0,0),$$

which implies, by (13) and the previous expression, that

$$v_S(u,0) - v_S(0,0) = -\frac{\lambda u}{\lambda\,\mathbb{E}\left[D\right] + e^{\lambda\tau}}\, c_S.$$

3.2° Assume now that $u = 0$ and $0 \leq t \leq \tau$. For the value function at state $(0, 0)$, we clearly have

$$\begin{aligned} v_S(0,0) &= \mathbb{E}\left[\int_0^t \lambda e^{-\lambda s}\left(-s r_S + v_S(X,0)\right) ds\right] + e^{-\lambda t}(-t r_S + v_S(0,t)) \\ &= -\frac{1-e^{-\lambda t}}{\lambda} r_S + (1-e^{-\lambda t})\mathbb{E}\left[v_S(X,0)\right] + e^{-\lambda t} v_S(0,t). \end{aligned}$$

By (13) and the result of 3.1°, we get, after some manipulations,

$$v_S(0,t) - v_S(0,0) = \frac{e^{\lambda t} - 1}{\lambda\,\mathbb{E}\left[D\right] + e^{\lambda\tau}}\, c_S,$$

which completes the proof. □

6 Experiments

In this section, we present the results of a series of simulation experiments designed to evaluate the gains of combining dispatching control (Section 4.1) with the lookahead policies for switching off and turning on servers (Sections 4.2 and 4.3, respectively). We compare this combined control approach with performing dispatching control only, to quantify the value of the lookahead policies. In addition to investigating the potential gains, we also explore the relative benefits of the two lookahead approaches.

The first experiment (Table 1), serving as a baseline for the remaining experiments, was performed under the following parameter settings: $d = 10$, X exponentially distributed with rate 1, and Λ was chosen such that the resulting loads $\rho = \Lambda/(2\mu)$ in Table 1 were achieved. The cost parameters were $e = 10$, $\gamma = 0.6$, $c_S = 100$, $c_T = 10$, and $c_P = 1$. The average cost rate with dispatching control and both lookahead policies is denoted by r_{LA}, while the average cost rate with dispatching control only is denoted by r_D. Simulations were run for 1000000 simulated time units. At the lowest load ($\rho = 0.2$), adding the lookahead

Table 1. Average cost rates for the first experiment

ρ	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
r_D	14.50	22.83	34.46	46.24	55.89	65.98	84.99	140.47
r_{LA}	18.47	22.78	28.31	35.41	45.11	58.78	82.07	137.05

resulted in a higher average cost rate. At first glance, this is counterintuitive, as additional control possibilities should decrease the cost. The issue here is that the lookahead for server turnoffs is too aggressive in keeping the server on – the dispatching control and this lookahead are designed separately and at low loads they appear to actually counteract each other. This effect was seen in varying degrees in all of the experiments. The best gain is 30.6 percent at $\rho = 0.5$.

The second experiment (Table 2) was the same as Experiment 1, but c_T was reduced to 1. Here, the problematic behavior at lower loads seen in the first

Table 2. Average cost rates for the second experiment

ρ	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
r_D	8.48	10.39	14.58	19.00	21.80	23.66	26.11	32.13
r_{LA}	8.50	15.71	17.34	19.22	21.19	23.19	25.81	31.95

experiment is more pronounced. This is due to energy costs being the dominant part of the average cost rate. The fact that the lookahead policy often keeps the server on is even more disadvantageous, as leaving the server on can only negatively impact the average energy cost rate.

The third experiment (Table 3) was the same as the first, but c_T was increased to 20. The key observation for this experiment is that the most significant gains are seen at moderate loads (a maximum gain of 41.5 percent at $\rho = 0.5$). This can be explained by the fact that at moderate loads, when the server that is always on is the only server operating, long queue lengths develop so that the other server is required. However, the server that can be switched off is then idle at a high frequency. Thus, it appears that both lookahead policies would be of value. The reality is that the lookahead to turn the server off was the only

Table 3. Average cost rates for the third experiment

ρ	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
r_D	21.13	35.96	56.04	75.61	92.20	111.80	151.48	258.47
r_{LA}	23.02	30.48	40.35	53.43	71.72	98.90	143.63	261.10

mechanism that was used – the lookahead to turn the server on was used at most once in each run. This was true of all experiments in this section.

The fourth experiment (Table 4) was the same as the first, but c_T was increased to 1000. Here, the gain of including lookahead is amplified, as the average

Table 4. Average cost rates for the fourth experiment

ρ	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
r_D	663.03	1290.46	2088.64	2854.21	3559.25	4611.84	6624.41	11992.76
r_{LA}	469.35	787.80	1220.39	1828.96	2683.30	4006.41	6268.40	11946.07

cost rate is almost completely determined by the average holding cost rate. The maximum gain is 71.1 percent at $\rho = 0.4$.

The fifth experiment (Table 5) was the same as the first, but X was chosen to follow a hyperexponential distribution with two phases with means 0.01 and 100 (the overall mean was 1). Here, the maximum gain is 15.7 percent at $\rho = 0.5$. The

Table 5. Average cost rates for the sixth experiment

ρ	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
r_D	38.12	84.52	185.51	372.18	710.54	1483.53	2899.18	6835.29
r_{LA}	34.34	76.79	171.93	321.70	699.95	1343.16	2797.43	6647.91

presence of very large jobs (high variance of service times) appears to mitigate the gains. The mechanism for this is not obvious, but one possibility is that as large jobs can be sent to both servers, the fact that we are using the Random Routing policy as our base policy for dispatching is problematic – size-aware routing may be a better choice.

The sixth experiment (Table 6) was the same as the first, but X was chosen to be constant. Here, the maximum gain is 15.2 percent at $\rho = 0.5$. The reduced variance leads to less opportunity for improvement, potentially due to the decreased variability of the workload at the server that is always on – there are no large fluctuations that require the additional control. Combining the observations from the sixth and seventh experiments, we see that the opportunities for improvement diminish as the service time variance approaches very small or very large values.

The seventh experiment (Table 7) was the same as the first, but d was reduced to 1. The maximum improvement is 13.7 percent at $\rho = 0.5$. The fact that the maximum improvement has decreased is not surprising, as the short setup times mean that the penalty paid for poor turnoff decisions is not as severe.

Table 6. Average cost rates for the seventh experiment

ρ	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
r_D	12.94	18.94	31.01	47.30	59.55	70.98	86.71	118.52
r_{LA}	12.94	23.63	30.84	41.06	54.01	68.65	86.03	119.54

Table 7. Average cost rates for the eighth experiment

ρ	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
r_D	14.32	21.17	29.03	36.74	44.46	54.36	71.29	119.26
r_{LA}	14.31	22.12	26.75	32.31	39.29	49.34	66.74	118.88

The eighth experiment (Table 8) was the same as the first, but d was increased to 100. The maximum gain is 22.2 percent, at $\rho = 0.5$. The gains are generally

Table 8. Average cost rates for the ninth experiment

ρ	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
r_D	14.50	23.53	49.18	111.34	169.28	212.47	249.52	325.44
r_{LA}	19.81	27.39	44.28	91.15	168.31	212.01	249.41	324.69

lower as the routing control tends to keep the server that can be switched off busy at all times (thus avoiding the long setup times), so there are less opportunities for the lookahead to be used.

In summary, the simulation results suggest that:

1. The lookahead for server turnoffs is the mechanism for reducing the cost.
2. The most gain from dynamic switch-off control is made at "moderate" values of load, service time variability, and setup times. Outside of these values, dispatching control alone appears to be sufficient.
3. When energy costs are dominant, dispatching control also appears to be sufficient.

Note that dispatching decisions indirectly control also the energy consumption due to the assumed default configuration where Server 1 was NeverOff and Server 2 InstantOff, explaining the latter two observations.

7 Conclusions

We considered the joint problem of combining dispatching control and server sleep state control, computing near optimal policies using one step of policy iteration for dispatching and lookahead techniques for server sleep state control. In addition to providing these policies explicitly, we identified when this joint control approach is most effective. Some issues for future work:

1. Is it possible to quantify the gap between state-dependent and state-independent sleep state control? This would give insight into the value of state information in making these control decisions. The work in [21] suggests that this

value goes to zero in a many server asymptotic regime, but the answer to this question for finite systems is of interest. Answering this question would involve characterizing (near) optimal state-independent sleep state control.
2. How does the approach scale? One important related question is determining which servers are always on.
3. If the servers are not FCFS can similar gains be expected?
4. For high variance service time distributions, it may be useful to consider a different initial policy for the dispatching control problem. One possibility is a SITA-like policy [3].

References

1. Barroso, L., Hölzle, U.: The case for energy-proportional computing. IEEE Computer 40, 12, 33–37 (2007)
2. Chen, Y., Das, A., Qin, W., Sivasubramaniam, A., Wang, Q., Gautam, N.: Managing server energy and operational costs in hosting centers. In: Proc. of ACM Sigmetrics 2005, pp. 303–314 (2005)
3. Crovella, M., Harchol-Balter, M., Murta, C.: Task assignment in a distributed system: Improving performance by unbalancing load. In Proc. of ACM Sigmetrics 1998, pp. 268–269 (1998)
4. Gandhi, A., Doroudi, S., Harchol-Balter, M., Scheller-Wolf, A.: Exact analysis of the M/M/k/setup class of Markov chains via recursive renewal reward. In: Proc. of ACM Sigmetrics 2013, pp. 153–166 (2013)
5. Gandhi, A., Gupta, V., Harchol-Balter, M., Kozuch, M.: Optimality analysis of energy-performance trade-off for server farm management. Performance Evaluation 67, 1155–1171 (2010)
6. Gandhi, A., Harchol-Balter, M., Adan, I.: Server farms with setup costs. Performance Evaluation 67, 1123–1138 (2010)
7. Gandhi, A., Harchol-Balter, M., Raghunathan, R., Kozuch, M.: AutoScale: Dynamic, robust capacity management for multi-tier data centers. ACM Transactions on Computer Systems 30, 14:1–14:26 (2012)
8. Gebrehiwot, M., Aalto, S., Lassila, P.: Optimal sleep-state control of energy-aware M/G/1 queues. In: Proc. of ValueTools 2014, pp. 82–89 (2014)
9. Gebrehiwot, M., Aalto, S., Lassila, P.: Energy-performance trade-off for processor sharing queues with setup delay. Operations Research Letters 44, 101–106 (2016)
10. Gebrehiwot, M., Aalto, S., Lassila, P.: Energy-aware server with SRPT scheduling: analysis and optimization. In: Proc. of QEST 2016, pp. 107–122 (2016)
11. Harchol-Balter, M.: Performance Modeling and Design of Computer Systems: Queueing Theory in Action. Cambridge University Press (2013)
12. Hyytiä, E., Penttinen, A., Aalto, S.: Size- and state-aware dispatching problem with queue-specific job sizes. European Journal of Operational Research 217, 357–370 (2012)
13. Hyytiä, E.: Lookahead actions in dispatching to parallel queues. Performance Evaluation 70, 859–872 (2013)
14. Hyytiä, E., Righter, R., Aalto, S.: Task assignment in a heterogeneous server farm with switching delays and general energy-aware cost structure. Performance Evaluation 75-76, 17–35 (2014)
15. Hyytiä, E., Righter, R., Aalto, S.: Energy-aware job assignment in server farms with setup delays under LCFS and PS. In: Proc. of ITC 26 (2014)

16. Maccio, V., Down, D.: On optimal policies for energy-aware servers. In: Proc. of IEEE MASCOTS 2013, pp. 31–39 (2013)
17. Maccio, V., Down, D.: On optimal control for energy-aware queueing systems. In: Proc. of ITC 27, pp. 98–106 (2015)
18. Maccio, V., Down, D.: Exact analysis of energy-aware multiserver queueing systems with setup times. In: Proc. of IEEE MASCOTS 2016, pp. 11–20 (2016)
19. Mitrani, I.: Service center trade-offs between customer impatience and power consumption. Performance Evaluation 68, 1222–1231 (2011)
20. Mitrani, I.: Managing performance and power consumption in a server farm. Annals of Operations Research 202, 121–134 (2013)
21. Mukherjee, D., Dhara, S., Borst, S., Leeuwaarden, J.: Optimal service elasticity in large-scale distributed systems. In: Proc. of ACM Sigmetrics 2017 (2017).
22. Penttinen, A., Hyytiä, E., Aalto, S.: Energy-aware dispatching in parallel queues with on-off energy consumption. In: Proc. of IEEE IPCCC 2011 (2011)
23. Phung-Duc, T.: Multiserver queues with finite capacity. In: Proc. of ASMTA 2015, pp. 173–187 (2015)
24. Phung-Duc, T.: Exact solutions for M/M/c/Setup queues. Telecommunication Systems 64, 309–324 (2017)
25. Puterman, M.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley (1994)
26. Slegers, J., Thomas, N., Mitrani, I.: Dynamic server allocation for power and performance. In: Proc. of SIPEW 2008, pp. 247–261 (2008)
27. Welch, P.: On a generalized M/G/1 queuing process in which the first customer of each busy period receives exceptional service. Operations Research 12, 736–752 (1964)
28. Whittle, P.: Optimal Control: Basics and beyond. Wiley (1996)