Renting Edge Computing Resources for Service Hosting

Aadesh Madnaik(✉), Sharayu Moharir, and Nikhil Karamchandani

Indian Institute of Technology Bombay, Mumbai, India
aadesh.madnaik@iitb.ac.in, {sharayum,nikhilk}@ee.iitb.ac.in

Abstract. We consider the setting where a service is hosted on a third-party edge server deployed close to the users and a cloud server at a greater distance from the users. Due to the proximity of the edge servers to the users, requests can be served at the edge with low latency. However, as the computation resources at the edge are limited, some requests must be routed to the cloud for service and incur high latency. The system's overall performance depends on the rent cost incurred to use the edge server, the latency experienced by the users, and the cost incurred to change the amount of edge computation resources rented over time. The algorithmic challenge is to determine the amount of edge computation power to rent over time. We propose a deterministic online policy and characterize its performance for adversarial and stochastic i.i.d. request arrival processes. We also characterize a fundamental bound on the performance of any deterministic online policy. Further, we compare the performance of our policy with suitably modified versions of existing policies to conclude that our policy is robust to temporal changes in the intensity of request arrivals.

Keywords: Service hosting · edge computing · competitive ratio

1 Introduction

Software as a Service (SaaS) instances like search engines, online shopping platforms, navigation services, and Video-on-Demand services have recently gained popularity. Low latency in responding to user requests/queries is essential for most of these services. This necessitates the use of edge resources in a paradigm known as edge-computing [24], i.e., storage and computation power close to the resource-constrained users, to serve user queries. Due to limited computation resources at the edge, such services are often also deployed on cloud servers which can serve requests that cannot be served at the edge, albeit with more latency given the distance between the cloud servers and the users. Ultimately, introducing edge-computing platforms facilitates low network latency coupled with higher

This work is supported by a SERB grant on Leveraging Edge Resources for Service Hosting.

E. Hyytiä and V. Kavitha (Eds.): VALUETOOLS 2022, LNICST 482, pp. 277–291, 2023.
https://doi.org/10.1007/978-3-031-31234-2_17

computational capabilities. For instance, consider a scenario where a child goes missing in an urban setting [25]. While cameras are widely used for security, it is challenging to leverage the information as a whole because of privacy and data traffic issues. In the edge computing paradigm, a workaround would be to push a request to search for the child to a certain subset of devices, thereby making the process faster and more efficient than cloud computing. Several other avenues of edge computing exist in the forms of cloud offloading, AR/VR-based infotainment, autonomous robotics, Industry 4.0 and the Internet of Things (IoT). Several industry leaders offer services for edge resources, e.g., Amazon Web Services [3], Oracle Cloud Infrastructure [13] and IBM with 5G technology [12].

This work considers a system with cloud servers and third-party-owned edge servers. Edge resources, i.e., storage and computation power, can be rented via short-term contracts to host services. Storage resources are needed to store the code, databases, and libraries for the service and computation resources are required to compute responses to user queries. As edge servers are limited in computational capabilities, there is a cap on the number of concurrent requests that can be served at the edge [26]. The amount of edge computational resources rented for the service governs the number of user requests that can be served simultaneously at the edge. We focus on a service that is hosted both on the cloud and edge servers, and the amount of edge computational resources rented can be changed over time based on various factors, including the user request traffic and the cost of renting edge computation resources. Service providers provision for elasticity in the quantity of edge resources rented, and the clients can exploit this based on the number of request arrivals [18]. The total cost incurred by the system is modelled as the sum of the rent cost incurred to use edge resources, the cost incurred due to high latency in serving requests that have to be routed to the cloud, and the switching cost incurred every time the amount of edge computation resource rented is changed [30]. The algorithmic challenge in this work is to determine the amount of edge computation resources to rent over time in the setting where the request arrival sequence is revealed causally with the goal of minimizing the overall cost incurred.

1.1 Our Contributions

We propose a deterministic online policy called Better-Late-than-Never (BLTN) inspired by the RetroRenting policy proposed in [21] and analyze its performance for adversarial and, i.i.d. stochastic request arrival patterns. In addition to this, we also characterize fundamental limits on the performance of any deterministic online policy for adversarial arrivals in terms of competitive ratio against the optimal-offline policy. Further, we compare the performance of BLTN with a suitably modified version of the widely studied Follow the Perturbed Leader (FTPL) policy [5,19] via simulations. Our results show that while the performance of BLTN and FTPL is comparable for i.i.d. stochastic arrivals, for arrival processes with time-varying intensity, e.g., a Gilbert-Elliot-like model, BLTN significant outperforms FTPL. The key reason for this is that BLTN puts extra emphasis on recent arrival patterns of making decisions, while FTPL uses the

entire request arrival history to make decisions. For all settings under consideration, the simulations demonstrate that BLTN differs little in performance from the optimal online policy despite not having information about the incoming request arrival process.

1.2 Related Work

There has been a sharp increase in mobile application latency and bandwidth requirements, particularly when coupled with time-critical domains such as autonomous robotics and the Internet of Things (IoT). These changes have ushered in the advent of the edge computing paradigm away from the conventional remote servers, as discussed in the surveys [1,15,23]. The surveys alongside several academic works elaborate on and model the dynamics of such systems. We briefly discuss some relevant literary works.

Representations of the problem considered in [6,26] model the decision making of which services to cache and which tasks to offload as a mixed-integer nonlinear optimization problem. In these cases, the problem is NP-hard. Similarly, [8] models the problem as a graph colouring problem and solves it using parallel Gibbs sampling. While these works try to solve a one-shot cost-minimization problem, in this work we consider the dynamic nature of decision-making based on the input request sequence.

Another model considered in [28] for service hosting focused on the joint optimization of service hosting decision and pricing is a two-stage interactive game between a base-station that provides pricing for edge servers and user equipment which decides whether to offload the task. In another game-theoretic setup, [14,29] delve into the economic aspects of edge caching involving interactions amongst different stakeholders. Some heuristic algorithms have been employed in the works [2,10]. Their approach for the problem is through resource constraint in the latency from the view-point of the edge-cloud infrastructure and not the application provider.

Stochastic models of the system have been considered in [9,17,27]. While [27] assumes that the underlying requests follow a Poisson process, [9,17] do not make any prior assumptions regarding the same. [9,17], through Contextual Combinatorial Multiarmed Bandits aim to use a learning-based approach to make decisions. [27] formulates the service migration problem as a Markov decision process (MDP) to design optimal service migration policies. These models do not provide any worst-case guarantees for the algorithm, simply average guarantees. In our work, we aim to provide both performance guarantees which are crucial to sensitive applications with large variations in the arrival patterns.

Closest to our work, [20–22] consider the setting where a service is always hosted at the cloud and consider the algorithmic task of determining when to host the service at the edge server as a function of the arrival process and various system parameters. The key difference between our work and [20–22] is that, in [20–22], once the service is hosted at the edge, the amount of edge computation resources available for use by the service is either fixed or effectively unlimited. Our model allows us to choose the level of computation resources to rent

which is a feature available in popular third-party storage/computation resource providers like AWS and Microsoft Azure. Another critical difference between our model and the [20–22] is the fact that we consider the setting where a non-zero switch cost is incurred every time we change the level of computation resource rented. Contrary to this, in [20–22], switch cost is unidirectional, i.e., a switch cost is incurred only when a service is not hosted at the edge in a time-slot and has to be fetched from the cloud servers to host on the edge server. Due to this, the algorithms proposed in [20–22] and their performance analyses do not directly extend to our setting. Other works on the service hosting problem include [31]. At a high level, our work is related to the rich body of work on caching [4,5,7,19].

2 Setting

We study a system consisting of a cloud server and a third-party-owned edge server. We focus on the problem of efficiently using edge resources from the perspective of a specific service provider. This service is hosted both at the edge and on the cloud server. Each user query/request is routed either to the edge or the cloud and the answer to the query is computed at that server and communicated back to the user, thus necessitating computation power both at the edge and at the cloud servers. We consider a time-slotted setting where the amount of edge computation power rented by the service provider can be changed over time via short-term contracts.

Request Arrival Process: We consider adversarial and stochastic arrivals. Under the adversarial setting, we make no structural assumptions on the number of requests arriving over time. For our analytical results for stochastic arrivals, we consider the setting where arrivals are i.i.d. over time.

Assumption 1 *(i.i.d. stochastic arrivals). Let X_t be the number of requests arriving in time-slot t. Then, for all t, $\mathbb{P}(X_t = x) = p_x$ for $x = 0, 1, 2, \cdots$.*

In the Gilbert-Elliot-like Model, we make the following assumption:

Assumption 2 *(Gilbert-Elliot (GE) Model). Using [11] as a basis, we consider an arrival process governed by a two-state Markov chain, A_H and A_L. Transitions from state $A_H \rightarrow A_L$ and from state $A_L \rightarrow A_H$ occur with probabilities p_{HL} and p_{LH}. The state transition diagram has been described in Fig. 1. We refer to the two states as the high state and the low state. Under the GE model, if the Markov chain is in the high state, the requests arrive as Poisson(λ_H), and they are Poisson(λ_L) otherwise.*

Sequence of Events in Each Time-Slot: We first have request arrivals. These requests are served by the edge server subject to constraints due to limited computation power at the edge. The remaining requests, if any, are forwarded to the cloud server for service. The system then makes a decision on how much edge computation power to rent for the next time-slot.

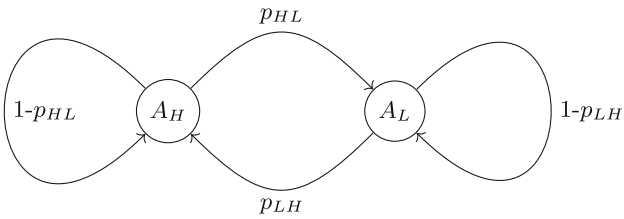


Fig. 1. Gilbert-Elliot Model as a Markov Chain

The algorithmic challenge is to determine how much edge computation power to rent over time. Let $\mathcal{P}$ be a candidate policy that determines the amount of computation power rented by the service provider over time.

2.1 Cost Model and Constraints

We build on the assumptions in [20–22]. Under policy $\mathcal{P}$, the service provider incurs three types of costs.

- *Rent cost* ($C^{\mathcal{P}}_{R,t}$): The service provider can choose one of two possible levels of edge computation power to rent in each time-slot, referred to as high (H) and low (L). The rent cost incurred per time-slot for levels H and L are denoted by c_H and $c_L (< c_H)$ respectively.
- *Service cost* ($C^{\mathcal{P}}_{S,t}$): This is the cost incurred due to the latency in service user requests. Given the proximity of the edge servers and the users, no service cost is incurred for requests served at the edge. A cost of one unit is levied on each request forwarded to the cloud server. The highest number of requests that can be served at the edge at edge computation power levels H and L are denoted by κ_H and $\kappa_L (< \kappa_H)$ respectively.
- *Switch cost* ($C^{\mathcal{P}}_{W,t}$): Switching from edge computation power level H to L and L to H results in a switch cost of W_{HL} and W_{LH} units respectively.

The number of requests that can be served by the edge server in a time-slot is limited to κ_H for state S_H and κ_L for state S_L in $\mathbb{Z}^+$, where $\mathbb{Z}^+$ is the set of all positive integers. Let $r_t \in \{H, L\}$ denote the edge computation power rented during time-slot t and X_t denote the number of request arrivals in time-slot t. It follows that

$$C^{\mathcal{P}}_t = C^{\mathcal{P}}_{R,t} + C^{\mathcal{P}}_{S,t} + C^{\mathcal{P}}_{W,t}, \tag{1}$$

$$\text{where, } C^{\mathcal{P}}_{R,t} = \begin{cases} c_H & \text{if } r_t = H \\ c_L & \text{if } r_t = L \end{cases}$$

$$C^{\mathcal{P}}_{S,t} = \begin{cases} X_t - \min\{X_t, \kappa_H\} & \text{if } r_t = H \\ X_t - \min\{X_t, \kappa_L\} & \text{if } r_t = L \end{cases}$$

$$C^{\mathcal{P}}_{W,t} = \begin{cases} W_{HL} & \text{if } r_{t-1} = H \text{ and } r_t = L \\ W_{LH} & \text{if } r_{t-1} = L \text{ and } r_t = H \\ 0 & \text{otherwise.} \end{cases}$$

Remark 1. We limit our discussion to the case where $\kappa_H - \kappa_L > c_H - c_L$. If $\kappa_H - \kappa_L \leq c_H - c_L$, the optimal policy is to always use computation level L.

2.2 Performance Metrics

We use the following metrics for adversarial and stochastic request arrivals.

For adversarial arrivals, we compare the performance of a policy $\mathcal{P}$ with the performance of the optimal offline policy (OPT-OFF) which knows the entire arrival sequence a priori. The performance of policy $\mathcal{P}$ is characterized by it competitive ratio $\rho^{\mathcal{P}}$ defined as

$$\rho^{\mathcal{P}} = \sup_{a \in \mathcal{A}} \frac{C^{\mathcal{P}}(a)}{C^{\text{OPT-OFF}}(a)}, \tag{2}$$

where $\mathcal{A}$ is the set of all possible finite request arrival sequences, and $C^{\mathcal{P}}(a)$ and $C^{\text{OPT-OFF}}(a)$ are the total costs of service for the request arrival sequence a under the policy $\mathcal{P}$ and the optimal offline policy respectively.

For i.i.d. stochastic arrivals, we compare the performances of a policy $\mathcal{P}$ with the optimal online policy (OPT-ON) which might know the statistics of the arrival process, but does not know the sample path. The performance metric $\sigma^{\mathcal{P}}_T$ is defined as the ratio of the expected cost incurred by policy $\mathcal{P}$ in T time-slots to that of the optimal online policy in the same time interval. Formally,

$$\sigma^{\mathcal{P}}(T) = \frac{\mathbb{E}\left[\sum_{t=1}^{T} C^{\mathcal{P}}_t\right]}{\mathbb{E}\left[\sum_{t=1}^{T} C^{\text{OPT-ON}}_t\right]}, \tag{3}$$

where $C^{\mathcal{P}}_t$ is as defined in (1).

Goal: The goal is to design online policies with provable performance guarantees for both adversarial and stochastic arrivals.

3 Policies

In our analysis, we focus the discussion towards *online* policies. At each time-slot, a singular decision must be made determining whether to switch states.

3.1 Better Late Than Never (BLTN)

The BLTN policy is inspired by the RetroRenting policy proposed in [21]. BLTN is a deterministic policy that uses recent arrival patterns to evaluate decisions by checking if it made the correct choice in hindsight. Let $t_{\text{switch}} < t$ be the most recent time when the state was changed from $H \rightarrow L$ or $L \rightarrow H$ under BLTN. The policy searches for a time-slot τ such that $t_{\text{switch}} < \tau < t$, and the total cost incurred is lower if the state is switched in time-slot $\tau - 1$ and switched back in time-slot t than the cost incurred if the state is not changed during time-slots $\tau - 1$ to t. If there exists such a time τ, BLTN switches the state in time-slot t.

Consider a scenario where the state in time slot t is S_H. Let $t_{\text{switch}} < t$ be the time when the server had last changed state to S_L under BLTN. Let H_i and L_i denote cost incurred in time slot i where H_i and L_i are evaluated for $r_i = r_{i-1} = H$ and L respectively. Analytically, the decision to switch to state S_L is made if the algorithm can find a time τ such that

$$W_{LH} + W_{HL} + \sum_{t_{\text{switch}} \leq i < \tau} H_i + \sum_{\tau \leq j \leq t} L_j < \sum_{t_{\text{switch}} \leq i \leq t} H_i$$

$$\text{which simplifies to } W_{LH} + W_{HL} < \sum_{i=\tau}^{t} (H_i - L_i). \tag{4}$$

A similar analytical condition can be made for the decision to switch from S_L to state S_H. The decision is made if the algorithm can find a time-slot τ such that

$$W_{LH} + W_{HL} < \sum_{i=\tau}^{t} (L_i - H_i). \tag{5}$$

A naive implementation of the algorithm has been constructed in the Appendix of [16].

While a naive implementation of the BLTN policy can have $\mathcal{O}(T)$ space and time complexity, using techniques proposed in [21], the time and computational complexity can be reduced to $\mathcal{O}(1)$ as shown through Algorithm 1.

3.2 Follow the Perturbed Leader (FTPL)

FTPL [5,19] is a randomized policy. In time-slot t, it compares suitably perturbed versions of the cost incurred from time 1 to t under two static decisions, i.e., state L from time 1 to t and state H from time 1 to t. The state of the system is then set to the one which has the lower perturbed cost.

The variation of the perturbation $\mathcal{N}(0, \sqrt{t})$ increases as $\sqrt{t}$, while the total difference in cost scaled linearly with time. This implies that the FTPL policy, over time, chooses to remain static in the state with the least cost.

Algorithm 1: Better Late than Never (BLTN)

Input: Sum of switch costs W units, maximum number of our service requests served by edge server (κ_H and κ_L), rent cost: c_H and c_L, number of requests: x_t, $\underline{x}_t^H = \min\{x_t, \kappa_H\}$, $\underline{x}_t^L = \min\{x_t, \kappa_L\}$, $t > 0$
Output: Service hosting strategy $r_{t+1} \in \{H, L\}$, $t > 0$
Initialize: Service hosting variable $r_1 = 0$, $\Delta(0) = 0$
for ***each*** *time-slot* t **do**
  $\Delta(t-1) = \Delta(t)$
  if $r_t = H$ **then**
    $\Delta(t) = \max\left\{0, \Delta(t-1) + \underline{x}_t^L - \underline{x}_t^H + c_H - c_L\right\}$
    if $\Delta(t) > W$ **then**
      $t_{\text{switch}} = t$
      $\Delta(t) = 0$
      return $r_{t+1} = L$
    else
      return $r_{t+1} = H$
    end
  else if $r_t = L$ **then**
    $\Delta(t) = \max\left\{0, \Delta(t-1) + \underline{x}_t^H - \underline{x}_t^L + c_L - c_H\right\}$
    if $\Delta(t) > W$ **then**
      $t_{\text{switch}} = t$
      $\Delta(t) = 0$
      return $r_{t+1} = H$
    else
      return $r_{t+1} = L$
    end
  end
end

Algorithm 2: Follow the Perturbed Leader (FTPL)

Input: Switch costs W_{HL} and W_{LH} units, maximum number of our service requests served by edge server (κ_H and κ_L), rent cost: c_H and c_L, number of requests: x_t, $\underline{x}_t^H = \min\{x_t, \kappa_H\}$, $\underline{x}_t^L = \min\{x_t, \kappa_L\}$, $t > 0$, Gaussian distribution with mean μ and variance σ: $\mathcal{N}(\mu, \sigma)$
Output: Service hosting strategy $r_{t+1} \in \{H, L\}$, $t > 0$
Initialize: Service hosting variable $r_1 = 0$, $\Delta(0) = 0$
for ***each*** *time-slot* t **do**
  $\Delta(t-1) = \Delta(t)$
  $\Delta(t) = \Delta(t-1) + \underline{x}_t^L - \underline{x}_t^H + c_H - c_L$
  if $\Delta(t) + \gamma\mathcal{N}(0, \sqrt{t}) > 0$ **then**
    return $r_{t+1} = L$
  else
    return $r_{t+1} = H$
  end
end

3.3 An Illustration

We consider a sample sequence of arrivals. Let r_t be the state sequence with time index t. We consider the case where $W_{LH} = W_{HL} = 275, c_H = 600, c_L = 400, \kappa_H = 700, \kappa_L = 300$ and a request sequence

Number of requests:	900	900	900	900	900	900	200	200	200	200	200
Time-slot index:	1	2	3	4	5	6	7	8	9	10	11

Initially, we consider the edge server to be in state S_L. We observe that the optimal state to serve 900 incoming requests is S_H. While hosting under the BLTN policy, the first switch to state S_H occurs in the time-slot 3, thus $r_4 = H$ and $r_{1,2,3} = L$. The cost incurred in the case where the state is S_L till $t = 3$ is $\sum_{l=1}^{3}(x_l - \kappa_L)^+ + (3 - 1 + 1) \times c_L = 3000$, while the cost incurred in state S_H is $\sum_{l=1}^{3}(x_l - \kappa_H)^+ + (3 - 1 + 1) \times c_H = 2400$. The difference in the cost equates $600 > W = 550$, and that is the first time the condition 5 is satisfied. t_{switch} is updated to 3, and $r_4 = H$.

From time-slot 4 onward, BLTN hosts in state S_H up to time-slot 8. We set $\tau = 6 > t_{switch}$ and evaluate the condition 4. Setting $t = 8, \tau = 6$, we have $\sum_{l=6}^{8}((x_l - \kappa_H)^+ - (x_l - \kappa_L)^+) + (8 - 6 + 1) \times (c_H - c_L) = 600 \geq W = 550$. This is the first time-slot since 3 that the condition is satisfied, thus $r_8 = H, r_{9,\ldots} = L$ till the next time BLTN decides to switch.

4 Main Results and Discussion

Our first theorem characterizes the performance of BLTN for adversarial arrivals by giving a worst-case guarantee on the performance of the BLTN policy against the optimal offline policy. We also characterize a lower bound performance of any deterministic online policy.

Theorem 1. *Let* $\Delta\kappa = \kappa_H - \kappa_L$, $\Delta c = c_H - c_L$, $W = W_{LH} + W_{HL}$. *If* $\Delta\kappa > \Delta c$ *then,*

(a) $\rho^{BLTN} \leq \left(1 + \frac{2W + \Delta\kappa}{W\left(1 + \frac{c_H}{\Delta\kappa - \Delta c} + \frac{c_L}{\Delta c}\right)}\right)$,

(b) $\rho^{\mathcal{P}} \geq min\left\{\frac{\Delta\kappa + c_L}{c_H}, \frac{c_H}{c_L}, \frac{\Delta\kappa + c_L + c_H + W}{c_H + c_L + W}\right\}$, *for any deterministic policy* $\mathcal{P}$.

Theorem 1 provides a worst-case guarantee on the performance of the BLTN policy against the optimal offline policy. Unlike the BLTN policy, the optimal offline policy has complete information of the entire arrival sequence beforehand. We note that the competitive ratio of BLTN improves as the sum of switch costs $(W_{LH} + W_{HL})$ increases. Also, the competitive ratio of BLTN increases linearly with the difference of the caps on requests served at the edge, $\Delta\kappa$. Supplementing it, we have Theorem 1 (b) which shows that the competitive ratio of *any* deterministic online policy increases linearly with $\Delta\kappa$. While Theorem 1 (a) provides a worst-case guarantee for the BLTN policy, it must be noted

that through subsequent simulations, the performance of BLTN is substantially closer to the optimal offline policy.

Next we summarize the performance of the BLTN policy for i.i.d. stochastic arrivals (Assumption 1, Sect. 2).

This lemma gives a bound on the expected difference between the costs incurred in a time-slot by the BLTN policy and the optimal online policy. We use the functions $f(\cdot), g(\cdot)$ which are defined in the Appendix of [16]. The functions $f(\cdot), g(\cdot)$ are formulated using Hoeffding's inequality to bound the probability of certain events. We use the functions $f(\cdot), g(\cdot)$ for the sake of compactness.

Lemma 1. *Let* $\Delta_t^{\mathcal{P}} = \mathbb{E}[C_t^{\mathcal{P}} - C_t^{OPT\text{-}ON}]$, $\mu_H = \mathbb{E}[\underline{X}_{t,H}]$, $\mu_L = \mathbb{E}[\underline{X}_{t,L}]$, $\Delta\mu = \mu_H - \mu_L$, $\Delta\kappa = \kappa_H - \kappa_L$, *and* $\Delta c = c_H - c_L$.

$$f(\Delta\kappa, \lambda, W, \Delta\mu, \Delta c) = (W + \Delta\mu - \Delta c) \times \left(2 \left\lceil \frac{\lambda W}{\Delta\mu - \Delta c} \right\rceil \frac{\exp(-2\frac{(\Delta\mu - \Delta c)^2 \frac{W}{\Delta c}}{(\Delta\kappa)^2})}{1 - \exp(-2\frac{(\Delta\mu - \Delta c)^2}{(\Delta\kappa)^2})} + \exp(-2\frac{(\lambda-1)^2 W(\Delta\mu - \Delta c)}{\lambda(\Delta\kappa)^2}) \right), \text{ and}$$

$$g(\Delta\kappa, \lambda, W, \Delta\mu, \Delta c) = (\Delta c - \Delta\mu + W) \times \left(\exp(-2\frac{(\lambda-1)^2(\Delta c - \Delta\mu)W}{\lambda(\Delta\kappa)^2}) + 2 \left\lceil \frac{\lambda W}{\Delta c - \Delta\mu} \right\rceil \frac{\exp(-2\frac{(\Delta c - \Delta\mu)^2 \frac{W}{\Delta\kappa - \Delta c}}{(\Delta\kappa)^2})}{1 - \exp(-2\frac{(\Delta c - \Delta\mu)^2}{(\Delta\kappa)^2})} \right).$$

Then, under Assumption 1,

- *Case* $\Delta\mu > \Delta c$:

$$\Delta_t^{BLTN}(\lambda) \leq \begin{cases} W + \Delta\mu - \Delta c, & t \leq \left\lceil \frac{\lambda W}{\Delta\mu - \Delta c} \right\rceil \\ f(\Delta\kappa, \lambda, W, \Delta\mu, \Delta c), & t > \left\lceil \frac{\lambda W}{\Delta\mu - \Delta c} \right\rceil \end{cases}.$$

- *Case* $\Delta\mu < \Delta c$:

$$\Delta_t^{BLTN}(\lambda) \leq \begin{cases} W + \Delta c - \Delta\mu, & t \leq \left\lceil \frac{\lambda W}{\Delta c - \Delta\mu} \right\rceil \\ g(\Delta\kappa, \lambda, W, \Delta\mu, \Delta c), & t > \left\lceil \frac{\lambda W}{\Delta c - \Delta\mu} \right\rceil \end{cases}.$$

We can conclude that for large enough t, the difference between the cost incurred by BLTN and the optimal online policy in time-slot t, decays exponentially with W and $|\Delta\mu - \Delta c|$.

Theorem 2 gives an upper bound on the ratio of the expected cost incurred under BLTN and the optimal online policy in a setting where request arrivals are stochastic. In the statement of this theorem, we used the functions f and g which are defined in Lemma 1.

Theorem 2. *Let* $\nu = \mathbb{E}[X_t]$, $\mu_H = \mathbb{E}[\underline{X}_{t,H}]$, *and* $\mu_L = \mathbb{E}[\underline{X}_{t,L}]$. *Let the rent cost per time-slot be* c_H *or* c_L *depending on the states* S_H *or* S_L *respectively. Define* $\Delta\mu = \mu_H - \mu_L$, $\Delta\kappa = \kappa_H - \kappa_L$, $\Delta c = c_H - c_L$. *Recall the definition of* $\sigma_T^{\mathcal{P}}$ *given in* (3).

– *Case $\Delta\mu > \Delta c$: For the function f defined in Lemma 1,*

$$\sigma^{BLTN}(T) \leq \min_{\lambda>1} \left(1 + \frac{\lceil \frac{\lambda W}{\Delta\mu - \Delta c} \rceil (W + \Delta\mu - \Delta c)}{T(\nu - \mu_H + c_H)} + \frac{\left(T - \lceil \frac{\lambda W}{\Delta\mu - \Delta c} \rceil\right) f(\Delta\kappa, \lambda, W, \Delta\mu, \Delta c)}{T(\nu - \mu_H + c_H)}\right),$$

– *Case $\Delta\mu < \Delta c$: For the function g defined in Lemma 1,*

$$\sigma^{BLTN}(T) \leq \min_{\lambda>1} \left(1 + \frac{\lceil \frac{\lambda W}{\Delta c - \Delta\mu} \rceil (W + \Delta c - \Delta\mu)}{T(\nu - \mu_L + c_L)} + \frac{\left(T - \lceil \frac{\lambda W}{\Delta c - \Delta\mu} \rceil\right) g(\Delta\kappa, \lambda, W, \Delta\mu, \Delta c)}{T(\nu - \mu_L + c_L)}\right).$$

We observe that the bounds in Lemma 1 worsen with an increase in $\Delta\kappa$. It must be noted that this is a bound obtained using Hoeffding's inequality which does not assume any specific i.i.d. process (Chernoff bound presents a stronger inequality here). For generic cases, the performance of BLTN does not worsen with an increase in $\Delta\kappa$ as shown via simulations in the next section.

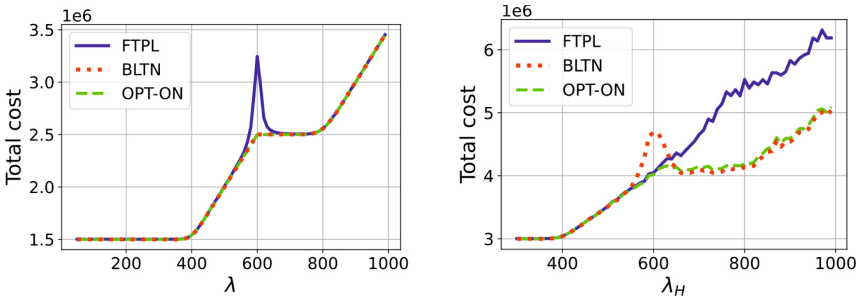
For large values of T, the bound on the ratio of total expected costs reduces exponentially with the sum of switch costs W. The performance guarantees obtained for BLTN in this section show that BLTN performs well in both the general and the i.i.d. stochastic settings without making any assumptions on the request arrival process.

5 Simulations

Since our analytical results only provide bounds on the BLTN policy, we now compare the performance of BLTN, FTPL, and the optimal online policy which uses the knowledge of the statistics of the arrival process to make decisions via simulations. Recall that both BLTN and FTPL do not know the statistics of the arrival process.

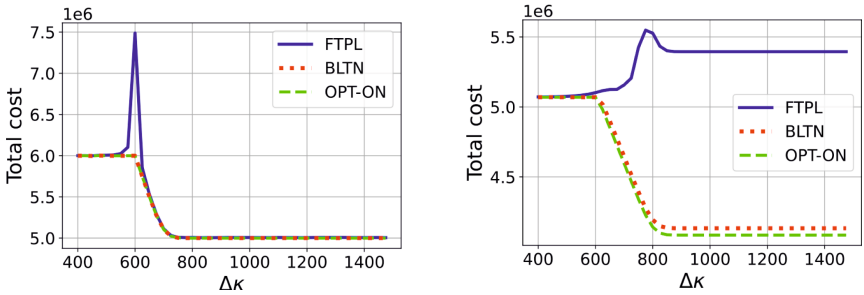
Unless stated otherwise, the parameter values in the simulations are as follows: $c_L = 300$; $\Delta c = 300$; $k_L = 400$; $\Delta\kappa = 400$; $W_{HL} = W_{LH} = 300$. For the GE model, the transition probability from either state of the two-state Markov chain is 0.01. In Algorithm 2, we set $\gamma = 500$, through empirical observations of overall performance. All the simulations have been averaged over the same set of 50 random seeds over 10,000 time-slots. The captions highlight the arrival sequence model - Assumption 1 (i.i.d. Poisson) or Assumption 2 (GE).

In Fig. 2, we see that for i.i.d. Poisson arrivals, the performance of BLTN matches that of FTPL for most of the λ values considered except around $\lambda = 600$. At this value of λ, the expected cost incurred at levels L and H is very close and as a result, the switch cost incurred by FTPL is high, thus leading to poor performance. We note that for the GE model, BLTN outperforms FTPL for a large range of λ_H. The superior performance of BLTN is a consequence of the fact that unlike FTPL, BLTN puts added emphasis on recent arrival patterns when making decisions. The same trend follows in Figs. 3, 4, 5 and 6.



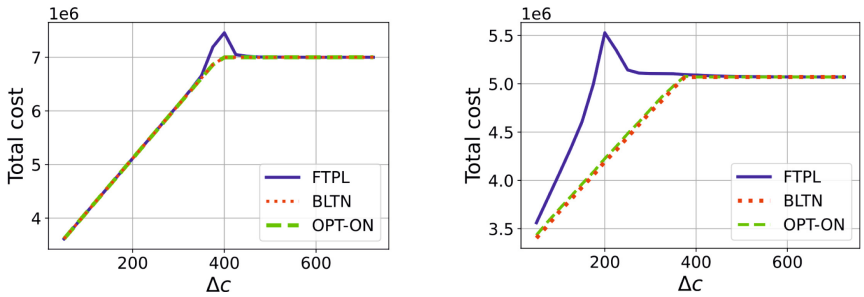
(a) i.i.d. Poisson (b) GE

Fig. 2. Performance of various policies as a function of the request arrival rate. For the GE model, we fix $\lambda_L = 300$ and vary λ_H.



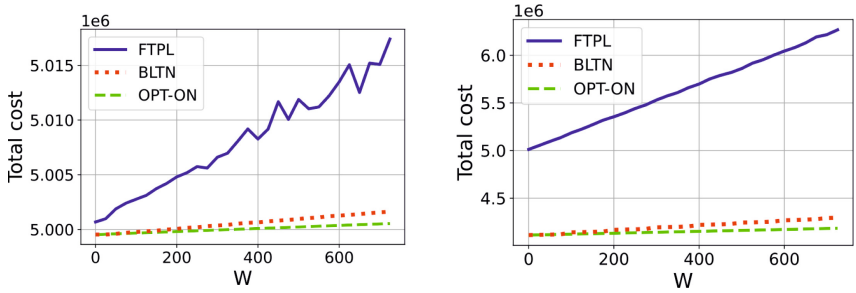
(a) i.i.d. Poisson, $\lambda = 700$ (b) GE, $\lambda_H = 800$, $\lambda_L = 300$

Fig. 3. Performance of various policies as a function of $\Delta\kappa$



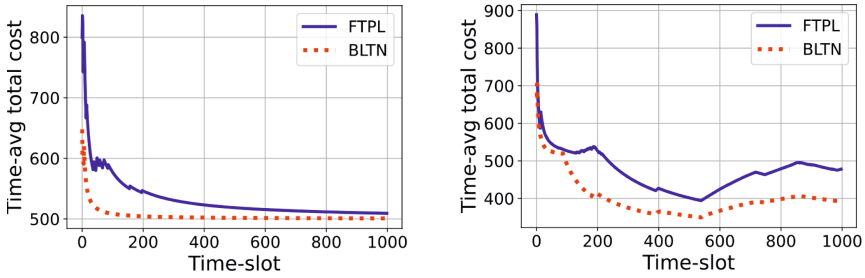
(a) Poisson($\lambda = 700$) (b) $GE(\lambda_H = 800, \lambda_L = 300)$

Fig. 4. Performance of various policies as a function of difference in rent costs Δc



(a) Poisson($\lambda = 700$) (b) $GE(\lambda_H = 800, \lambda_L = 300)$

Fig. 5. Performance of various policies as a function of switch costs, $W = W_{HL} = W_{LH}$



(a) Poisson($\lambda = 700$) (b) $GE(\lambda_H = 800, \lambda_L = 300)$

Fig. 6. Time averaged total cost

Through Theorems 1 and 2, it is suggested that the bounds worsen as $\Delta\kappa$ increases. However, under Assumption 1, there is significant difference in performance only when $\Delta\kappa = \Delta c$, and under Assumption 2, there is significant difference whenever $\Delta\kappa > \Delta c$.

In arrival sequences characterized by Assumption 2, usually BLTN performs better owing to its ability to draw conclusions from history. FTPL fails to account for switching costs and so, is sub-optimal.

6 Conclusion

We consider the problem of renting edge computing resources for serving customer requests at the edge. We propose an online policy called Better-Late-Than-Never (BLTN) and provide performance guarantees for adversarial and stochastic request arrivals. Further, we compare the performance of BLTN with the widely studied FTPL policy. We conclude that BLTN outperforms FTPL for most settings considered, especially when the statistics of the arrival process are time-varying. The main reason for this is that BLTN makes decisions based on recent request arrival patterns while FTPL uses the entire request arrival history to make decisions.

References

1. Abbas, N., Zhang, Y., Taherkordi, A., Skeie, T.: Mobile edge computing: a survey. IEEE Internet Things J. **5**(1), 450–465 (2018). https://doi.org/10.1109/JIOT.2017.2750180
2. Ascigil, O., Tasiopoulos, A.G., Phan, T.K., Sourlas, V., Psaras, I., Pavlou, G.: Resource provisioning and allocation in function-as-a-service edge-clouds. IEEE Trans. Serv. Comput. **15**(4), 2410–2424 (2022). https://doi.org/10.1109/TSC.2021.3052139
3. AWS (2022). https://aws.amazon.com
4. Belady, L.A.: A study of replacement algorithms for a virtual-storage computer. IBM Syst. J. **5**(2), 78–101 (1966)
5. Bhattacharjee, R., Banerjee, S., Sinha, A.: Fundamental limits on the regret of online network-caching. Proc. ACM Meas. Anal. Comput. Syst. **4**(2), 1–31 (2020)
6. Bi, S., Huang, L., Zhang, Y.J.A.: Joint optimization of service caching placement and computation offloading in mobile edge computing system. arXiv preprint arXiv:1906.00711 (2019)
7. Borst, S., Gupta, V., Walid, A.: Distributed caching algorithms for content distribution networks. In: 2010 Proceedings IEEE INFOCOM, pp. 1–9. IEEE (2010)
8. Chen, L., Xu, J.: Collaborative service caching for edge computing in dense small cell networks. arXiv preprint arXiv:1709.08662 (2017)
9. Chen, L., Xu, J.: Budget-constrained edge service provisioning with demand estimation via bandit learning. arXiv preprint arXiv:1903.09080 (2019)
10. Choi, H., Yu, H., Lee, E.: Latency-classification-based deadline-aware task offloading algorithm in mobile edge computing environments. Appl. Sci. **9**(21), 4696 (2019)
11. Gilbert, E.N.: Capacity of a burst-noise channel. Bell Syst. Techn. J. **39**(5), 1253–1265 (1960). https://doi.org/10.1002/j.1538-7305.1960.tb03959.x
12. IBM (2022). https://www.ibm.com/cloud/edge-computing
13. Infrastructure, O.C. (2022). https://www.oracle.com/a/ocom/docs/cloud/edge-services-100.pdf
14. Jiang, C., Gao, L., Wang, T., Luo, J., Hou, F.: On economic viability of mobile edge caching. In: ICC 2020–2020 IEEE International Conference on Communications (ICC), pp. 1–6. IEEE (2020)
15. Luo, Q., Hu, S., Li, C., Li, G., Shi, W.: Resource scheduling in edge computing: a survey. IEEE Commun. Surv. Tutor. **23**(4), 2131–2165 (2021). https://doi.org/10.1109/COMST.2021.3106401
16. Madnaik, A., Moharir, S., Karamchandani, N.: Renting edge computing resources for service hosting (2022). https://doi.org/10.48550/ARXIV.2207.14690, https://arxiv.org/abs/2207.14690
17. Miao, Y., Hao, Y., Chen, M., Gharavi, H., Hwang, K.: Intelligent task caching in edge cloud via bandit learning. IEEE Trans. Netw. Sci. Eng. **8**(1), 625–637 (2020)
18. Mouradian, C., Naboulsi, D., Yangui, S., Glitho, R.H., Morrow, M.J., Polakos, P.A.: A comprehensive survey on fog computing: State-of-the-art and research challenges. IEEE Commun. Surv. Tutor. **20**(1), 416–464 (2017)
19. Mukhopadhyay, S., Sinha, A.: Online caching with optimal switching regret. In: 2021 IEEE International Symposium on Information Theory (ISIT), pp. 1546–1551. IEEE (2021)
20. Narayana, V.C.L., Agarwala, M., Karamchandani, N., Moharir, S.: Online partial service hosting at the edge. In: 2021 International Conference on Computer Communications and Networks (ICCCN), pp. 1–9. IEEE (2021)

21. Narayana, V.C.L., Moharir, S., Karamchandani, N.: On renting edge resources for service hosting. ACM Trans. Model. Perform. Eval. Comput. Syst. **6**(2), 1–30 (2021)
22. Prakash, R.S., Karamchandani, N., Kavitha, V., Moharir, S.: Partial service caching at the edge. In: 2020 18th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOPT), pp. 1–8. IEEE (2020)
23. Puliafito, C., Mingozzi, E., Longo, F., Puliafito, A., Rana, O.: Fog computing for the internet of things: A survey. ACM Trans. Internet Technol. **19**(2), 18:1–18:41 (2019). https://doi.org/10.1145/3301443, http://doi.acm.org/10.1145/3301443
24. Satyanarayanan, M.: The emergence of edge computing. Computer **50**(1), 30–39 (2017)
25. Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L.: Edge computing: vision and challenges. IEEE Internet Things J. **3**(5), 637–646 (2016). https://doi.org/10.1109/JIOT.2016.2579198
26. Tran, T.X., Chan, K., Pompili, D.: COSTA: cost-aware service caching and task offloading assignment in mobile-edge computing. In: 2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), pp. 1–9. IEEE (2019)
27. Wang, S., Urgaonkar, R., Zafer, M., He, T., Chan, K., Leung, K.K.: Dynamic service migration in mobile edge computing based on Markov decision process. IEEE/ACM Trans. Netw. **27**(3), 1272–1288 (2019). https://doi.org/10.1109/TNET.2019.2916577
28. Yan, J., Bi, S., Duan, L., Zhang, Y.-J.A.: Pricing-driven service caching and task offloading in mobile edge computing. IEEE Trans. Wirel. Commun. **20**(7), 4495–4512 (2021). https://doi.org/10.1109/TWC.2021.3059692
29. Zeng, F., Chen, Y., Yao, L., et al.: A novel reputation incentive mechanism and game theory analysis for service caching in software-defined vehicle edge computing. Peer-to-Peer Netw. Appl. **14**, 467–481 (2021). https://doi.org/10.1007/s12083-020-00985-4
30. Zhang, M., Zheng, Z., Shroff, N.B.: An online algorithm for power-proportional data centers with switching cost. In: 2018 IEEE Conference on Decision and Control (CDC), pp. 6025–6032 (2018). https://doi.org/10.1109/CDC.2018.8619443
31. Zhao, T., Hou, I.H., Wang, S., Chan, K.: RED/LED: an asymptotically optimal and scalable online algorithm for service caching at the edge. IEEE J. Sel. Areas Commun. **36**(8), 1857–1870 (2018)