On the Regret of Online Edge Service Hosting

R Sri Prakash
IIT Bombay
prakash.14191@gmail.com

Nikhil Karamchandani
IIT Bombay
nikhilk@ee.iitb.ac.in

Sharayu Moharir
IIT Bombay
sharayum@ee.iitb.ac.in

ABSTRACT

We consider the problem of service hosting where a service provider can dynamically rent edge resources via short term contracts to ensure better quality of service to its customers. The total cost incurred by the system is modeled as a combination of the rent cost, the service cost incurred due to latency in serving customers, and the fetch cost incurred as a result of the bandwidth used to fetch the code/databases of the service from the cloud servers to host the service at the edge. In this paper, we compare multiple hosting policies with regret as a metric, defined as the difference in the cost incurred by the policy and the optimal policy over some time horizon T. In particular we consider the Retro Renting (RR) and Follow The Perturbed Leader (FTPL) policies proposed in the literature and provide performance guarantees on the regret of these policies. We show that under i.i.d Bernoulli arrivals, RR policy has linear regret while FTPL policy has constant regret. Next, we propose a variant of FTPL, namely Wait then FTPL (W-FTPL), which also has constant regret while demonstrating much better dependence on the fetch cost. We also show that under adversarial arrivals, RR policy has linear regret while both FTPL and W-FTPL have regret $O(\sqrt{T})$ which is order-optimal.

1. INTRODUCTION

Software as a Service (SaaS) instances like online navigation platforms, Video-on-Demand services, etc., have stringent latency constraints in order to provide good quality of experience to their customers. While most SaaSs use cloud resources, low latency necessitates the use of storage/computational resources at the edge, i.e., close to the end-user. A service is said to be hosted at the edge if the code and databases needed to serve user queries are stored on the edge servers and requests can be served at the edge.

We consider the setting where third-party resources can be rented via short-term contracts to host the service and the edge hosting status of the SaaS can be dynamically changed over time. If the service is not hosted at the edge, it can be fetched from the cloud servers to host at the edge by incurring a fetch cost. The performance of a hosting policy is a function of the rent cost, the fetch cost, and the quality of experience of the users. We refer to the algorithmic challenge of determining when to host the service at the edge as the *service hosting problem*.

Novel online service hosting policies with provable performance guarantees have been proposed in [4, 7]. The metric of interest in [4, 7] is the *competitive ratio*, defined as the ratio of the cost incurred by an online policy to the cost incurred by an offline optimal policy for the same request arrival sequence. Since the competitive ratio is multiplicative by definition, the absolute value of the difference in the cost incurred by an online policy and the optimal policy can be large even though the competitive ratio of the online policy is close to one. This motivates studying the performance of candidate policies in terms of *regret*, defined as the difference in the cost incurred by the policy and the offline optimal policy. Regret is a widely used metric in online learning [2], including recently for the caching problem [5, 1] which is closely related to the service hosting problem. In this work, one of our goals is to design hosting policies with provable guarantees in terms the regret. Another key dimension in the performance guarantees of online policies is the assumption made on the request arrival process. Commonly studied settings include the *stochastic setting* and the *adversarial setting* and we provide performance guarantees for both settings.

For the service hosting problem, [4] proposed the Retro-Renting (RR) policy and showed that it has a constant competitive ratio with the offline optimal policy. On the other hand, for the closely related caching problem, several policies inspired by the recent advances in online convex optimization have been proposed including Online Gradient Ascent [5], Online Mirror Descent [6] and Follow the Perturbed Leader (FTPL) [3, 1]. In particular, FTPL has been shown to have order-optimal regret for the caching problem in the adversarial setting [3, 1]. In this work, we study regret for the RR and FTPL policies for the service hosting problem, under both the stochastic and adversarial settings. Since RR is a deterministic policy, its regret performance in the adversarial setting is poor. A limitation of FTPL which is a randomized policy is that it makes hosting decisions agnostic of the fetch cost. As a result, in some cases, FTPL is prone to fetching and evicting the service multiple times in the initial time-slots when its estimate of the request arrival rate is noisy, thus leading to poor performance.

Our Contributions: We propose a variant of the FTPL policy called Wait then Follow the Perturbed Leader (W-FTPL). W-FTPL is a randomized policy that takes into account the fetch cost in its decision-making. More specifically, W-FTPL does not fetch the service for an initial wait period which depends on the request arrivals and is an increasing function of the fetch cost. Following the wait period, W-FTPL mimics the FTPL policy.

For i.i.d. Bernoulli request arrivals and the regret metric, we show that RR is sub-optimal and FTPL and W-FTPL order-optimal with respect to the time horizon. While the regret of FTPL can increase linearly with the fetch cost, the regret of W-FTPL increases at most logarithmically. The improved performance of W-FTPL over FTPL is a consequence of the fact that W-FTPL avoids most of the fetches made by FTPL in the initial time-slots and by the end of the wait period, its estimate of the arrival rate is accurate enough to avoid multiple fetches.

For the adversarial setting, we first characterize a fundamental lower bound on the regret of any online hosting policy. In terms of regret, we then show that RR is strictly suboptimal, while FTPL and W-FTPL have order-optimal performance with respect to time.

2. SETTING

We consider a system consisting of a back-end server and an edge-server. The back-end server always hosts the service and can serve any requests that are routed to it. In addition, the service can be hosted at the edge-server. If a service is hosted at the edge, requests can be served locally, i.e., at the edge. The hosting status at the edge can be changed over time. If the service is not hosted at the edge, it can also be fetched from the back-end server to host at the edge. We consider a time-slotted system.

Request arrivals: We consider two types of arrival processes. The first where arrivals are i.i.d. Bernoulli across time-slots with mean μ and the second where the arrival process is generated by an oblivious adversary[1] with at most one request arrival per slot.

Sequence of events in each time-slot: In each time-slot, we first make the service hosting decision for that time-slot. Following this, a request may arrive and is served either at the edge or by the back-end server.

Costs: We model three types of costs.
Rent cost: The system incurs a cost of $c \in (0,1)$ units per time-slot to host the service at the edge.[2]
Service cost: The system incurs a cost of 1 unit for each request that is served by the back-end server.
Fetch cost: The system incurs a cost of $M > 1$ units for each fetch of the service from the back-end server to host on the edge-server.

Let $r_t \in \{0,1\}$ denote the number of request arrivals in time-slot t and $\rho_t^{\mathcal{P}}$ denote the edge hosting status of the service in time slot t under policy $\mathcal{P}$, where $\rho_t^{\mathcal{P}} = 1$ if the service is hosted at the edge in time-slot t, and 0 otherwise. The total cost incurred in time-slot t by policy $\mathcal{P}$ denoted by $\mathcal{C}_t^{\mathcal{P}}(r_t)$ is the sum of the rent, service, and fetch costs. It follows that $\mathcal{C}_t^{\mathcal{P}}(r_t) = c\rho_t^{\mathcal{P}} + (1-\rho_t^{\mathcal{P}})r_t + M(\rho_t^{\mathcal{P}} - \rho_{t-1}^{\mathcal{P}})^+$. Let $r = \{r_t\}_{t\geq 1}$ denote the request arrival sequence and $\mathcal{C}^{\mathcal{P}}(T,r)$ denote the cumulative cost incurred by policy $\mathcal{P}$ in time-slots 1 to T. It follows that $\mathcal{C}^{\mathcal{P}}(T,r) = \sum_{t=1}^{T} \mathcal{C}_t^{\mathcal{P}}(r_t)$.

Performance metrics: For i.i.d. Bernoulli arrivals, the regret of a policy $\mathcal{P}$, denoted by $\mathcal{R}_B^{\mathcal{P}}(T)$, is defined as the expectation of the difference in the total cost incurred by the policy and the optimal static hosting policy. The optimal static hosting policy makes a hosting decision at $t = 1$ using the knowledge of the statistics of the request arrival process but not the entire sample-path. It follows that the expected cost incurred by the optimal static hosting policy in time-slots 1 to T is $\min\{cT + M, \mu T\}$, and therefore,

$$\mathcal{R}_B^{\mathcal{P}}(T) = \mathbb{E}_{\mathcal{P},r}[\mathcal{C}^{\mathcal{P}}(T,r)] - \min\{cT + M, \mu T\}.$$

For adversarial arrivals, the regret of a policy $\mathcal{P}$, denoted by $\mathcal{R}_A^{\mathcal{P}}(T)$, is defined as the expectation of the difference in the total cost incurred by the policy and the optimal static hosting policy in the worst case. It follows that the cost incurred by the optimal static hosting policy in time-slots 1 to T is $\min\{cT + M, \sum_{t=1}^{T} r_t\}$, and therefore,

$$\mathcal{R}_A^{\mathcal{P}}(T) = \sup_{r \in \mathcal{R}} \left(\mathbb{E}_{\mathcal{P}}[\mathcal{C}^{\mathcal{P}}(T,r)] - \min\{cT + M, \sum_{t=1}^{T} r_t\} \right).$$

Goal: The goal is to design online hosting policies with provable performance guarantees with respect to regret.

3. POLICIES

Our Policy: Our policy called Wait then Follow the Perturbed Leader (W-FTPL) is a variant of the popular FTPL policy. FTPL is a randomized policy and is known to perform well for the caching problem, in fact achieving orderwise optimal regret for adversarial arrivals [1]. At the end of each time-slot, FTPL retrospectively computes the costs that would have been incurred by the two static options: hosting the service throughout and not hosting the policy in any slot. It then perturbs the two costs by adding appropriately scaled version of independent samples from standard Gaussian random variables and chooses for the next time-slot the option with the lower value.

The key idea behind the W-FTPL policy is to not host the service for an initial wait period. This is to reduce the number of fetches made initially when estimate the arrivals is noisy. The duration of this wait period is a function of the arrival pattern seen till that time. Following the wait period, W-FTPL mimics the FTPL policy. Refer to Algorithm 1 for a formal definition of W-FTPL.

Algorithm 1: Wait then Follow The Perturbed Leader (W-FTPL)

```
Input: $c$, $M$, $\{\eta_t\}_{t\geq 1}$, $\beta$, $\{r_l\}_{l=1}^{t}$
1 Set $R_0 = 0$, $t = 1$, wait $= 1$, sample $\gamma_1, \gamma_2 \sim \mathcal{N}(0,1)$
2 for $t > 0$ do
3     $R_t = R_{t-1} + r_t$, wait $= \min\{\text{wait}, \mathbb{1}_{t < \frac{\beta \log M}{(c - R_t/t)^2}}\}$
4     $\rho_t = \mathbb{1}_{\text{wait} \neq 1} \times \mathbb{1}_{R_t + \eta_t\gamma_1 > ct + \eta_t\gamma_2}$
```

Retro-Renting (RR) [4]: The RR policy is a deterministic hosting policy. The key idea behind this policy is to use recent arrival patterns to make hosting decisions. We omit the details of this policy due to space constraints. The performance of RR with respect to the competitve ratio was analyzed in [4].

4. MAIN RESULTS AND DISCUSSION

In this section, we state and discuss our key results. Our first result characterizes the regret performance of the policies discussed in Section 3 for i.i.d. Bernoulli arrivals.

THEOREM 1. *Let the arrivals in each time-slot be i.i.d. Bernoulli with mean μ.*

[1]The entire request sequence is assumed to be fixed apriori.
[2]For $c \geq 1$ it is optimal to never host the service.

(a) $\mathcal{R}_B^{RR}(T) \geq M\left(\frac{T}{\frac{M}{1-c}+\frac{M}{c}+2}-1\right)\mu^{\lceil \frac{M}{1-c}\rceil}(1-\mu)^{\lceil \frac{M}{c}\rceil}$.

(b) $\mathcal{R}_B^{FTPL}(T) \leq \frac{16\alpha^2+2}{|c-\mu|^2}(M+|c-\mu|)$ *for* $\eta_t = \alpha\sqrt{t}$.

(c) $\mathcal{R}_B^{W\text{-}FTPL}(T) \leq 1 + \frac{\beta}{|c-\mu|} + \frac{\beta(1+4\log M)}{|c-\mu|^2} + \frac{\beta^2}{|c-\mu|^4}$ *for* $\eta_t = \alpha\sqrt{t}$ *and* $\beta = \max\{(1+4\alpha)^2, (1+\sqrt{2})^2\}$.

The key take-aways from Theorem 1 is that for i.i.d. Bernoulli arrivals, RR is strictly sub-optimal with respect to time and incurs linear regret while both FTPL and W-FTPL have constant regret with respect to time. Further, the upper bound on the regret of FTPL increases linearly with fetch cost M, while the upper bound on the regret of W-FTPL is proportional to $\log M$.

Note that FTPL makes hosting decisions agnostic of the fetch cost. As a result, FTPL changes the hosting status multiple times in the initial time-slots before eventually converging on the optimal hosting status. This leads to high switching costs. Compared to this, since W-FTPL does not change the hosting status in the first $O(\log M)$ time-slots, the switching cost incurred in the waiting period is zero. Further, once the waiting period is over, W-FTPL has enough information about the arrival process to make the right hosting decision with high probability, thus avoiding multiple fetch-evict cycles.

In Figure 1, 2, we compare the performance of the policies via simulations for i.i.d. Bernoulli arrivals with $c = 0.45$, $\mu = 0.4$, $\alpha = 0.1$, $\beta = 6$ averaged over 50 experiments. For Figure 1 we consider $M = 5$ and compare the regret as a function of the time horizon. We note that W-FTPL outperforms RR and FTPL policies and the results agree with Theorem 1. For Figure 2 we set $T = 5000$ and plot the total cost as a function of M. We observe that W-FTPL outperforms RR, FTPL and the key differentiating factor between the policies is the fetch cost incurred.

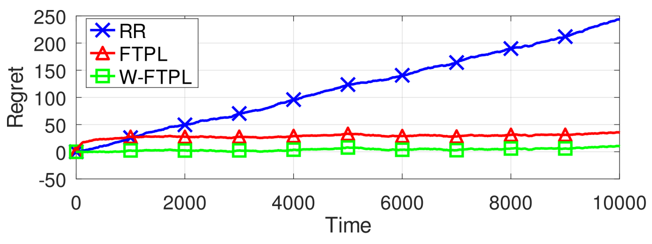

Figure 1: Regret as a function of time

Our second result characterizes the regret performance of the policies discussed in Section 3 and also provides a fundamental limit on the performance of any online policy for adversarial arrivals.

THEOREM 2. *If the arrivals are generated by an oblivious adversary under the constraint that at most one request arrives in each time-slot, then,*

(a) $\mathcal{R}_\mathcal{A}^{\mathcal{P}}(T) \geq \sqrt{\frac{Tc(1-c)}{2\pi}}\left[1 - \frac{1}{12(cT-1)(1-c)}\right]$ $\forall$ *policy* $\mathcal{P}$.

(b) $\mathcal{R}_A^{RR}(T) \geq M\left(\frac{T}{\frac{M}{1-c}+\frac{M}{c}+2}-1\right)$.

(c) $\mathcal{R}_A^{FTPL}(T) \leq \alpha\sqrt{2T\log 2} + 4(1+c)^2\frac{\sqrt{T}}{\alpha\sqrt{2\pi}} + \frac{3M}{\alpha\sqrt{\pi}}\sqrt{T}$ *for* $\eta_t = \alpha\sqrt{t}$ *for* $\alpha > 0$.

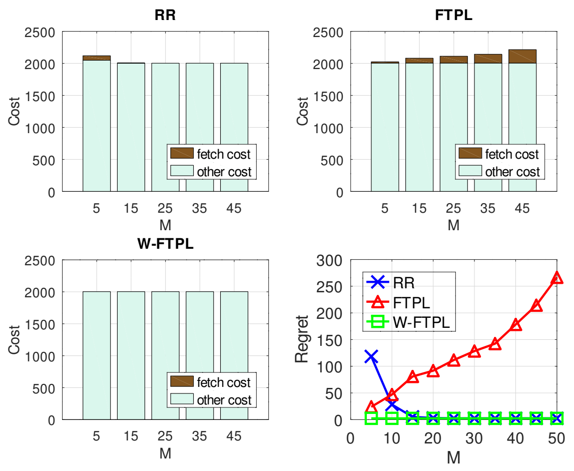

Figure 2: Cost, Regret as a function of fetch cost (M)

(d) $\mathcal{R}_A^{W\text{-}FTPL}(T) \leq \alpha\sqrt{2T\log 2} + 4(1+c)^2\frac{\sqrt{T}}{\alpha\sqrt{2\pi}} + \frac{3M}{\alpha\sqrt{\pi}}\sqrt{T} + \sqrt{\beta T\log M}$ *for* $\eta_t = \alpha\sqrt{t}$ *for* $\alpha > 0$ *and* $\beta > 0$.

The key take-away from Theorem 2 is that RR suffers linear regret while both FTPL and W-FTPL are order-optimal with respect to time. We thus conclude that for the regret metric, RR is strictly sub-optimal and W-FTPL performs well in both the adversarial and stochastic settings.

5. REFERENCES

[1] R. Bhattacharjee, S. Banerjee, and A. Sinha. Fundamental limits on the regret of online network-caching. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 4(2):1–31, 2020.

[2] T. L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

[3] S. Mukhopadhyay and A. Sinha. Online caching with optimal switching regret. In *2021 IEEE International Symposium on Information Theory (ISIT)*, pages 1546–1551. IEEE, 2021.

[4] V. C. L. Narayana, S. Moharir, and N. Karamchandani. On renting edge resources for service hosting. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, 6(2):1–30, 2021.

[5] G. S. Paschos, A. Destounis, L. Vigneri, and G. Iosifidis. Learning to cache with no regrets. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 235–243. IEEE, 2019.

[6] T. S. Salem, G. Neglia, and S. Ioannidis. No-regret caching via online mirror descent. In *ICC 2021-IEEE International Conference on Communications*, pages 1–6. IEEE, 2021.

[7] T. Zhao, I.-H. Hou, S. Wang, and K. Chan. Red/led: An asymptotically optimal and scalable online algorithm for service caching at the edge. *IEEE Journal on Selected Areas in Communications*, 36(8):1857–1870, 2018.