Proactive Serving Decreases User Delay Exponentially

Shaoquan Zhang
The Chinese University of Hong Kong

Longbo Huang
Tsinghua University

Minghua Chen
The Chinese University of Hong Kong

Xin Liu
University of California, Davis

ABSTRACT

In online service systems, the delay experienced by a user from the service request to the service completion is one of the most critical performance metrics. To improve user delay experience, recent industrial practice suggests a modern system design mechanism: *proactive serving*, where the system predicts future user requests and allocates its capacity to serve these upcoming requests proactively. In this paper, we investigate the fundamentals of proactive serving from a theoretical perspective. In particular, we show that proactive serving decreases average delay *exponentially* (as a function of the prediction window size). Our results provide theoretical foundations for proactive serving and shed light on its application in practical systems.

1. INTRODUCTION

The fast growing number of personal devices with Internet access, *e.g.*, smart mobile devices, has led to the blossom of diverse online service systems, such as cloud computing, cloud storage, online social networks, mobile Internet access, and a variety of online communication applications. In online service systems, delay experienced by a user from the service request to the service completion is one of the most critical performance metrics. For example, experiments at Amazon showed that every 100-millisecond increase in load time of Amazon.com would decrease revenues by 1% [5]. Google also found that an extra 0.5 seconds in search page generation time dropped traffic by 20% [5].

Traditionally, to reduce user delay and to improve quality of experience, a widely adopted design mechanism is *capacity boosting*, *i.e.*, increasing the service capacity by for example deploying more servers. However, such a mechanism may be expensive as it needs to provision for the peak demand and thus results in low average utilization due to the bursty nature of service requests, especially when the user arrivals are time-varying.

Recently, both industrial practice and academic studies suggest *proactive serving*, *i.e.*, serving future requests before they arrive, as a modern approach for reducing user delay. Proactive serving is based on the key observation on service request predictability. It is a technique that has been widely used in computer systems based on what is likely to happen next, such as cache pre-loading and command pre-fetching. Similarly, in cloud service systems, it is not atypical to have predictable service requests. For example, in cloud computing platforms, service jobs, such as indexing, page-ranking, backup, crawling, and maintenance-related load, are often predictable. In fact, in an industrial grade cloud computing system, we observe a significant portion of the workload to be periodic and thus predictable [4]. Intuitively, if one user is observed to watch football news consistently in the morning, then such contents can be preloaded in the future.

In practice, Amazon launched *Amazon Silk*, a mobile web browser in its tablet Kindle Fire [1]. All the web traffic from the browser goes through the Amazon cloud and gets managed by the servers in the cloud. Based on cached user traffic, the cloud uses machine learning techniques to predict what users will browse in the future. When service in the cloud is available, web pages that are likely to be requested are preloaded to users tablet by servers. In this manner, when the user clicks on the corresponding content, the loading is instant and the user delay is reduced to zero. This technique speeds up request responses and improves user browsing experience.

All the above exciting developments suggest proactive serving as a new design mechanism for reducing user delay: based on user request arrival prediction, the system can allocate its capacity proactively and pre-serve future requests to reduce the delay experienced by users. This observation naturally leads to a fundamental question: How much user delay reduction can we obtain by proactive serving?

In this paper, we explore the answer to the above open question and investigate the fundamentals of proactive serving from a queuing theory perspective. In particular, we study proactive serving with a prediction window of size ω, where one has the ability to predict future requests in a time window of ω and serve them if needed. We investigate the impact of proactive serving on reducing user delay as a function of the prediction window size ω.

In comparison, authors in [7] and [9] use future information to control admission of requests into the system to reduce waiting time. Previous studies also focus on request scheduling algorithm design based on future prediction but without proactive serving capacity [2], [6], [7]. The authors in [8] investigate how proactive serving reduces the probability of server outage. Predictive scheduling in controlled queuing systems is studied in [3], where Predictive Backpressure algorithm is proposed to achieve the optimal utility performance. In this paper, we study the capability of proactive serving in delay reduction and make the following contributions. First, we show that the average user delay decreases *exponentially* in the prediction window size ω under perfect prediction. Then, based on the insights, we prove that exponential delay reduction holds for proactive serving under imperfect prediction.

2. MODEL

Consider a service system as shown in Fig. 1. In the system, the single backend server provides service to incoming user requests that arrive at the system according to a continuous process $\{A(t)\}_t$ with rate λ. When a user request arrives and the server is idle, the request will be served.

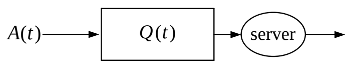

Figure 1: A single queue service system.

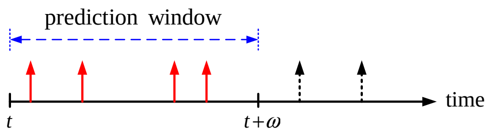

Figure 2: Prediction model.

Otherwise, the request waits in the queue $Q(t)$ for service. The queueing discipline is FCFS. Upon completing service, the request leaves the system. We assume that service times of requests are i.i.d. random variables with mean $1/\mu$. We define user delay as the time from when the user request arrives at the system till it leaves the system.

Now suppose that the service system has proactive serving capability. As shown in Fig. 2, we assume that the system can perfectly predict user request arrivals ω time ahead. That is, at time t, the system knows exactly in $(t, t+\omega)$ the request arrival epochs (red solid arrows) and the corresponding users who generate the requests. Meanwhile, we do not assume the knowledge of the service time of each user request.

Based on the arrival prediction, the system can allocate its service capacity to serve future requests proactively. Specifically, the servers can provide service to the users who are predicted to generate requests. The user requests that get pre-served will not enter the system. Such a proactive serving model captures service systems that can perform cache pre-loading or command pre-fetching. As a practical example of cache pre-loading, the web browser in Amazon's Kindle Fire can predict web page requests and pre-load desired web pages to users' tablets beforehand. When the user clicks on the predicted content, it gets the content immediately.

We depict the service system which can proactively serve future requests based on perfect prediction in Fig. 3. Let $Q_0(t)$ represent the queue of the requests that have arrived at the system and are waiting for service at time t, and $W_\omega(t)$ be the prediction window of size ω.

Each user request first goes through the prediction window $W_\omega(t)$ and then enters the queue $Q_0(t)$. The servers can serve the requests in both $Q_0(t)$ and $W_\omega(t)$. We remark that each request entering $W_\omega(t)$ will transit to $Q_0(t)$ after exactly ω amount of time, if it has not been pre-served before that. The requests will not queue up in $W_\omega(t)$. Thus $W_\omega(t)$ should be viewed as a pipe. User delay corresponds to the time that the request spends in $Q_0(t)$ and with the server, and it does not include the time spent in $W_\omega(t)$.

3. AVERAGE USER DELAY UNDER PERFECT PREDICTION

In this section, we assume user requests arrive according to a Poisson process and service times of requests are independent and identically distributed according to an exponential distribution. We characterize the average user delay of the system shown in Fig. 3. Let D^ω denote the user delay when the system can predict ω time ahead.

When $\omega = 0$, *i.e.*, without proactive serving, the system reduces to the classical M/M/1 queue. It's well known that the average user delay is given by

$$E\left[D^0\right] = \frac{1}{\mu - \lambda}. \tag{1}$$

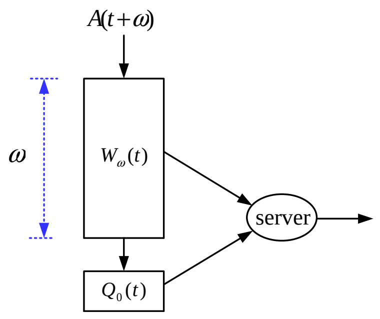

Figure 3: Service system with perfect prediction.

The probability density function of D^0 is also known as

$$f_{D^0}(t) = (\mu - \lambda)e^{-(\mu-\lambda)t}, t \geq 0. \tag{2}$$

To characterize the user delay, we first observe an interesting result that $Q^{sum}(t) \triangleq Q_0(t) + W_\omega(t)$ evolves the same as an M/M/1 system with a properly initialized queue. Based on this observation, the distribution of user delay under proactive serving, *i.e.*, D^ω, turns out to be a "shifted" version of that of user delay without proactive serving as shown in (2). Once we know the distribution of D^ω, we can compute the average user delay $E\left[D^\omega\right]$ as shown in the following theorem. Detailed proof can be found in [10].

Theorem 1. *Assume $\mu > \lambda$. The average user delay is given by*

$$E[D^\omega] = \frac{1}{\mu - \lambda} e^{-(\mu-\lambda)\omega}. \tag{3}$$

Theorem 1 reveals that the average user delay decays exponentially in the prediction window size ω. We have derived similar results for $G/G/1$ queue in [10]. These results indicates that a small amount of future information can improve user delay experience significantly. In particular, they suggests that, for Amazons new mobile web-browser described in Section 1, if the Amazon cloud can predict near-future web requests actually, the request response time can be reduced significantly.

4. AVERAGE USER DELAY UNDER IMPERFECT PREDICTION

In Section 3, we analyze the benefit of proactive serving under perfect arrival prediction. In this section, we study the performance of proactive serving under imperfect arrival prediction with two types of prediction errors. For ease of analysis and illustration, we consider a single server system. Due to the space limitation, we highlight only the main results in this section, and refer interested readers to our technical report [10] for more details.

4.1 Modeling

The first type of error is failing to predict actual arrivals, *i.e.*, miss detection (also called false negative). When miss detection happens, the arrival will be out of the system's vision. Therefore, it cannot be served proactively. Intuitively, such errors result in a "side flow" into the system and will affect the ultimate gain one can obtain by proactive serving. The other type of error is false alarm (also called false positive), which happens when the system mistakenly predicts the existence of non-existing arrivals. Such false arrivals will not eventually enter the system for service. However, the system may incorrectly allocate resources to serve them, resulting in wasted service opportunities.

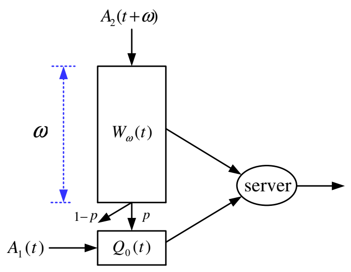

Figure 4: Service system with imperfect prediction.

We model the system with these two types of prediction errors by the model shown in Fig. 4. In this model, $\{A_2(t)\}_t$ represents the process of predicted arrivals, which include false alarms and actual arrivals that are predicted correctly. $\{A_1(t)\}_t$ instead represents the process of miss detections. $Q_0(t)$ stores requests that have already entered the system and are waiting for service at time t. $W_\omega(t)$ is the prediction window with length ω. Requests in $\{A_2(t)\}_t$ go through the prediction window and can be served proactively by the server. In contrast, requests in $\{A_1(t)\}_t$ enter $Q_0(t)$ directly and cannot be served proactively. False alarms in $\{A_2(t)\}_t$ will not enter $Q_0(t)$. They are either pre-served in $W_\omega(t)$ or depart once they leave the prediction window.

For tractability, we make the following two assumptions on $\{A_1(t)\}_t$ and $\{A_2(t)\}_t$. First, we assume that request arrivals in $\{A_2(t)\}_t$ are independent, and each request is an actual request with probability p and is a "false-alarm" request with probability $1-p$. The larger p is, the more accurate the prediction mechanism is. Second, we assume that $\{A_1(t)\}_t$ and $\{A_2(t)\}_t$ are independent Poisson processes. Note that these assumptions are reasonable for systems with a large number of users, where consecutive arrivals are likely originated from different users. In this case, the prediction about one arriving request can be considered as independent from others.

Denote $E[A_1(t)] = \lambda_1$ and $E[A_2(t)] = \lambda_2$. Since all the miss detections and the actual arrivals among the predicted arrivals compose the real arrivals, we have

$$\lambda_1 + p\lambda_2 = \lambda. \tag{4}$$

In this system, the server applies the FCFS service policy. Different from the case of perfect prediction, here we assume that the requests that are already in $Q_0(t)$ and the arrivals from $\{A_1(t)\}_t$ have preemptive priority. That is, the service of arrivals in $W_\omega(t)$ will be suspended unless the queue is empty and there is no new arrival entering $Q_0(t)$ from $\{A_1(t)\}_t$.

4.2 Impact of Miss Detections

Now suppose that are only miss detections in the system, *i.e.*, $p = 1$ and $\lambda_1+\lambda_2 = \lambda$. In this case, it is conceivable that the delay improvement may be less significant as compared to the perfect-prediction case, because miss detection cannot be pre-served by the system. We have the following result.

Theorem 2. *Assume $\lambda = \lambda_1+\lambda_2 < \mu$. The average user delay under miss detections is given by:*

$$\begin{aligned}&E[D^\omega] \qquad (5)\\ &= \frac{\lambda_1}{(\mu-\lambda_1)\lambda} + \frac{\lambda_2}{\lambda}\Bigg[\frac{\lambda^2-\lambda_1\mu}{\lambda_2^2(\mu-\lambda)} \cdot e^{\frac{\lambda_2(\lambda-\mu)}{\lambda}\omega} \cdot \mathbf{1}_{\lambda^2>\lambda_1\mu} + \\ &\frac{1}{2\pi}\int_0^{4\sqrt{\lambda_1\mu}} \frac{(\mu-\lambda)\sqrt{x(4\sqrt{\lambda_1\mu}-x)}e^{(-(\sqrt{\mu}-\sqrt{\lambda_1})^2-x)\omega}}{\left((\sqrt{\mu}-\sqrt{\lambda_1})^2+x\right)^2\cdot\left(\lambda x+(\sqrt{\lambda_1\mu}-\lambda)^2\right)}dx\Bigg],\end{aligned}$$

which decreases exponentially in ω.

The result in (6) consists of two components. The first component does not depend on ω. This part is due to that miss detections enter $Q_0(t)$ directly without going through the prediction window and thus proactive serving cannot reduce this part of delay. The second component of the delay decreases exponentially with ω and diminishes as the system predicts sufficiently far, which we formally established in the proof in [10]. This exponential delay reduction is due to that the system can pre-serve the predicted arrivals in the prediction window $W_\omega(t)$.

4.3 Impact of False Alarms

When there exist only false alarms in the system, *i.e.*, $\lambda_1 = 0$ and $p\lambda_2 = \lambda$, the server capacity will be wasted if a false alarm is pre-served. As a result, the power of proactive serving will be affected compared to the perfect prediction scenario. Despite this effect, as we will see, proactive serving still provides significant delay improvement.

Theorem 3. *Assume $\lambda = p\lambda_2 < \mu$. The average user delay under the impact of false alarms is given by*

$$E[D^\omega] = \begin{cases} \frac{\mu-\lambda_2}{(\mu-\lambda)^2}\frac{1}{e^{(\mu-\lambda_2)\omega}-\frac{\lambda_2-\lambda}{\mu-\lambda}} & \lambda_2 \neq \mu \\ \frac{1}{(\mu-\lambda)[(\mu-\lambda)\omega+1]} & \lambda_2 = \mu \end{cases}. \tag{6}$$

The average delay decreases exponentially in ω when $\lambda_2 \neq \mu$.

From (6), it is not immediately clear that $E[D^\omega]$ decreases exponentially in ω when $\lambda_2 \neq \mu$. Instead, we show in the proof in [10] that $E[D^\omega]$ can be lower and upper bounded by exponential functions which decrease exponentially in ω.

5. CONCLUSIONS

In this paper, we investigate the fundamentals of proactive serving from a queuing theory perspective. We show that proactive serving decreases average delay exponentially (as a function of the prediction window size). Our results provide theoretical justification for practical use of proactive serving.

6. REFERENCES

[1] Kindle fire. `http://www.amazon.com/gp/product/b0051vvob2`.
[2] B. Coleman. Quality vs. performance in lookahead scheduling. In *JCIS*, 2006.
[3] L. Huang, S. Zhang, M. Chen, and X. Liu. When backpressure meets predictive scheduling. In *Proc. MobiHoc*, 2014.
[4] A. Khan, X. Yan, S. Tao, and N. Anerousis. Workload characterization and prediction in the cloud: A multiple time series approach. In *IEEE Network Operations and Management Symposium*, 2012.
[5] R. Kohavi and R. Longbotham. Online experiments: Lessons learned. *IEEE Computer*, 2007.
[6] M. Mandelbaum and D. Shabtay. Scheduling unit length jobs on parallel machines with lookahead information. *Journal of Scheduling*, 2011.
[7] J. Spencer, M. Sudan, and K. Xu. Queueing with future information. *arXiv:1211.0618*, 2012.
[8] J. Tadrous, A. Eryilmaz, and H. El Gamal. Proactive resource allocation: harnessing the diversity and multicast gains. *IEEE Trans. Information Theory*, 2013.
[9] K. Xu and C. W. Chan. Using future information to reduce waiting times in the emergency department. *under submission.*
[10] S. Zhang, L. Huang, M. Chen, and X. Liu. Effect of proactive serving on user delay reduction in service systems. Techniical Report, CUHK. `http://www.ie.cuhk.edu.hk/~mhchen/papers/proa_serv.tr.pdf`.