Computer Communications 125 (2018) 24–37

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Quantitative quality estimation of cloud-based streaming services

Fang Yu*[,1,a], Yat-wah Wan[b], Rua-huan Tsaih[a]

[a] Department of Management Information Systems, National Chengchi University, No.64, Sec. 2, Zhinan Rd., Taipei, Taiwan, ROC
[b] Graduate Institute of Logistics Management, National Dong Hwa University, No. 1, Sec. 2, Da Hsueh Rd., Hualien, Taiwan, ROC

ARTICLE INFO

Keywords:
Simulation
Quantitative analysis
Queueing theory
Online streaming services
Cloud computing
Service quality

ABSTRACT

Cloud-based streaming services, such as real-time streaming video and gaming services, have emerged as popular online Internet applications in recent years. Providing systematic quality estimation before (or after) launching these services has raised a significant challenge due to dynamic runtime status of servers, clients and the network environment. This paper proposes a queueing model for the cloud-based streaming service in which packet level dynamics are taken into consideration so that customer-affected performance can be estimated by a hybrid simulation approach. The simulation approach is particularly useful for cloud service providers to evaluate the service quality before launching the service. The analytical model has two parts: (1) the virtual-machine-level service queueing model along with the stationary closed-form expressions on the average number of customers, the average waiting time, and the average number of employed virtual machines (VMs), and (2) the microscopic model and the simulation procedure on the customer side that capture the lag time of streaming packets. The simulation procedure is derived based on the analytical model. The simulation results show how the service quality is affected by server and customer performances, providing the insight for cloud resource provision and client parameter settings.

1. Introduction

Cloud computing has grabbed the spotlight in IT industries. No matter in the form of SaaS (i.e., Software-as-a-Service), IaaS (i.e., Infrastructure-as-a-Service), or PaaS (i.e., Platform-as-a-Service), cloud-based services have become more popular because they streamline business processes and cut cost for companies. By outsourcing digital storage, software, hardware, and even whole application systems to cloud service providers, companies can become *thin clients* with minimal communicating and computing facilities on their sides. To access to services pre-agreed with a service provider, a company only needs to connect the clients on his side to a server of the service provider, who has already configured his centralized computing hardware into *virtual machines* (VMs), i.e., artificial partitions providing a computing environment on which operating systems and programs can run. As far as the company on the client side is concerned, a VM behaves as if a physical computer right with the company. Yet the partitioning, configuration, and operation rules of VMs can be easily changed, and the elasticity on the usage of cloud resources allows VMs to be cloned and reallocated according to dynamic needs.

A cloud-based service provider provides different types of service to multiple customers at the same time. While the amount of investment by the service provider is a sunk cost, the effective and efficient deployment of resources is essential to its service level to customers, and eventually to its profitability.

The amount of resources deployed to a particular service is dependent on the pre-agreed service level between a service provider and its customers. There are various types of service levels, varying in the amount of computation power, storage, and speed of data transfer. With thin clients and fluctuating internet traffic, customer satisfaction is especially important. Any contract renewal would be shaky if on average customers experience many *interruptions of service* (i.e., the VM often temporarily stalls within a customer connection session), long *lag time* (i.e., long time waiting for a VM to resume function), or both, in sessions connecting to servers of the service provider. Thus, before launching the service, based on the prior assessment on customer characteristics, such as the arrival rate of demands, and the distribution of the length of connection sessions, cloud service providers would like to predict:

- the number of VMs required for a given set of operation rules to configure the VMs;
- the estimates of service levels such as the mean number of interruptions of service and mean lag time for a customer within his

* Corresponding author.
E-mail address: yuf@nccu.edu.tw (F. Yu).
[1] soslab.nccu.edu.tw

https://doi.org/10.1016/j.comcom.2018.04.017
Received 14 October 2017; Received in revised form 7 April 2018; Accepted 27 April 2018
Available online 04 May 2018
0140-3664/

connection session as the controllable parameters of the cloud service vary.

While desirable to have, the amount of resources required and the resulted service levels are difficult to predict. Consider a simple set of operation rules to configure the system: All the resources of a cloud service are configured into working (i.e., active) VMs, and any incoming customer is directed to the VM with the least workload, or the least number of customers if the workloads of customers are unknown, where ties are broken arbitrarily. An inviting approach to solve the problem is to model the VMs as servers, subject to random arrivals of customers requesting random services. Even with classical assumptions for queueing models, such as Poisson arrivals and exponential service times (i.e., connection times), where all random quantities are independent from each other, there is no closed-form solution to the problem (Gupta et al. [22]).

One may opt for simulation, the powerful generic tool that theoretically can be applied to estimate performance measures for any stochastic problems. This approach does not work in practice. For simplicity consider a system with only 100 VMs. At any time, theoretically any number of these 100 VMs may be working, each with any number of customers. The feasible states of the system, i.e., the combinations of the number of working VMs together with the combinations of their customers, are simply too astronomical to defy any precise statistical estimates for the system performance measures.

It turns out that lacking closed-form solution and missing precision are not the main problems of the approach. More important, a conventional queueing model that treats VMs as servers serving customers does not capture reality: it fails to model the dynamics in the physical layer of cloud-based service system, and misses the interaction between the dynamics and the performance of the system in service levels. In the following paragraphs we first describe the physical layer of a cloud-based service system and then explain the problems of a conventional queueing model.

In a typical cloud-based service [52], to obtain cloud-based streaming service, a customer sends a request for an index page through his *local device* to the cloud-based service system. On receiving the request, the *dispatcher* of the system, an interface between the customers and the system, passes the request to the *monitor*, which is the central control of the system. The monitor is the control unit programmed to monitor the workload of active VMs, to assign the VM for a customer (i.e., an incoming request), to initiate idling VMs for high workload, and possibly to retire VMs for low workload. The precise logic of the monitor depends on how it is programmed. The URL of the *cloud server*, the VM for a customer, is sent from the monitor to the local device of the customer through the dispatcher. During the connection session of a customer, its cloud server may work in the *processor-sharing* mode, i.e., the cloud server provides service to multiple customers at the same time. The precise sharing mode of a cloud server depends on its programmed *time-division multiplexing scheme*. When the service turn of a customer comes, for the duration of time allocated to the customer, the cloud server codes data in *packets* and sends them through the *virtual path* set up temporarily between the cloud server and the client local device. Even for a fixed pair of server and client, such virtual paths can change rapidly with time so that packets sent at different times may go through different virtual paths. As a result of the physical structure of a cloud-based service system, the service level of a customer certainly depends on the operation rules on initiation and retirement of VMs, the time-division multiplexing scheme within VMs, the power of the cloud server to code and transmit data, the random Internet traffic for the packets, and the power the local device to decode data. Moreover, as packets are sent through different virtual paths subject to different Internet traffic, they may not arrive at the client local device at the correct order; the packets for a video cannot be shown in a client local device if some earlier packets have not arrived. The service levels of a customer, such as interruptions and lag time, are affected by all these factors.

In modeling VMs as servers and packets as customers, the resulted queueing model completely misses the dynamics in physical layers of the cloud-based system. All aforementioned factors that affecting the service levels of the cloud-based service system as discussed in the above paragraph are ignored. Moreover, the waiting times from any such models are the *experience of individual packets, not of customers*. As far as we know, there is no analytical model that overcomes these problems by explicitly considering the aforementioned factors.

In this paper we develop an analytical approach to explicitly consider the dynamics in the physical layer of a cloud-based system, and relates it to the experience of customers. As the situation is too complicated, there is a compromise between the generality of the model and the tractability of its results. In particular, we make the following assumptions on the operation rules of VMs and characteristics of customers. While restricting the generality, the assumptions are common and frequently appear in the literature.

1. The customer requests occur according to a stationary Poisson process of rate λ.
2. The *connection times* that the customers plan to connect to the cloud servers are i.i.d. $\mathscr{C} \sim \exp(\mu)$, independent of the request process.
3. A VM adopts the *round-robin* rule to serve customers, each turn serving a customer for a fixed duration called *quantum*denoted as Δ.
4. The VMs are programmed to initiate and retire in the following way: At any time only one VM, the *corresponding worker*, receives new requests. At any epoch when accepting a new request makes B customers at the corresponding worker, the monitor takes two actions. First, it informs the dispatcher not to send any futher request to the current corresponding worker, i.e., the current corresponding worker starts its retirement. It will join the idle pool of VMs when all its customers have finished their sessions. Second, the monitor activates one idle VM as the new corresponding worker and asks the dispatcher to send futher arrivals to this VM.

The Poisson arrival process in the first assumption is a widely-used approximation for a large collection of potential customers of the cloud-based service each requesting the service with very small probability but the mean number of requests does not change with time. In many real-life applications, the customer requests are time varying, possibly in a cyclic pattern varying with the hour of a day, the day of a week, and both. An interpretation of our assumption is that we adopt the *simple peak hour approximation* discussed in Green and Kolesar [20], i.e., we use a stationary Poisson process with the average peak hour rate. Such an approximation is the worst case analysis as there are off-peak periods. Fortunately, the procedure that we develop can be applied as an approximation for a heterogeneous Poisson demand process with the procedure applied to each interval of piecewise constant arrival rate. More discussion on approximation errors (or accuracy) and generalization using steady-state approximation for time-dependent performance can be found in [10,21,24,30,42].

The second assumption, while restrictive, appears in many classical queueing models. In order to develop a computationally tractable model that captures the many features in the physical layer of a cloud-based system, we rely on this classical assumption. Previous studies on viewing time of users in practice can be found in [15,16,31,50]. Chen et al. [15] show that users spend a lot of time browsing, and only occasionally (around 20% of the time) watching a video to its completion. By predicting the user departure behavior, an effective streaming strategy [16] is proposed to avoid the waste on bandwidth to achieve better service quality. Xu et al. [50] show that viewing time of streaming users fits a hyper-exponential distribution, where all the data could be separated into two categories (i.e., short and long), and could be inferred via a Bayesian approach with information from content providers. Li et al. [31] indicate a notable impact of mobile device type for abandonment rates over 9 million access logs collected from the PPTV live streaming system.

The third assumption is fairly close to reality. The CPU of a computer in general works in discrete time unit, which, for ease of reference, is called quantum Δ in this paper. Unless the tasks are of different priorities, the round-robin time multiplexing rule is very common. There are queueing analyses of the round-robin packet scheduling policy in computer system that try to capture the effect of multiple clients on a server, or a VM in our context (Sakata et al. [39], and Rasch [37], Silberschatz et al. [40]). However, these models do not model the dynamics in the physical layer as we do.

The fourth assumption is the truly key assumption developed specially in this paper. As shown in subsequent Section 3, together with other assumptions, the fourth assumption leads to closed-form expressions for stationary (i.e., steady-state) distribution of the corresponding worker, the mean time from initiation to complete retirement for a VM, and the mean number of working VMs in the cloud-based service system. Such system characteristics are essential for resource provision before launching a cloud-based system. Moreover, the stationary distribution of the corresponding worker eventually reveals the service levels of the system, as measured by the mean number of interruptions and mean lag time of a customer during his connection session.

Under these four assumptions, we develop a solution approach to deduce *exactly* the service levels defined for the experience of customers, with the dynamics of the physical layer incorporated in the derivation of the service levels. The solution approach is of two levels. The macroscopic level models the connection sessions of customers; it finds the mean number of the VMs required and the distribution of the number of customers at the corresponding worker by steady-state analysis (Section 3). The microscopic level deduces by *simulating* the physical layer of the cloud-based system to get the service levels for the experience of customers (Section 4). The results include the mean number of interruptions and mean lag time of a customer as a function of the length of quantum Δ and upper bound of customer B. The four assumptions enable us skipping the infeasible task of simulating the whole system. Instead, we can simply repeatedly simulate the life cycles of *one* VM, from initiation as a corresponding worker, to retiring, and finally to idling. In these simulated life cycles, we trace the sample paths of *k-customers* who on arrival finds k-customers being served at the corresponding worker. The averages from these sample paths give the mean number of interruptions and the mean lag time in a connection session of a k-customer. With known stationary distribution of the number of customers at the corresponding worker, which matches with those of the k-customers, the mean number of interruptions and mean lag time of k-customers give the corresponding values of the system service levels. As our solution approach combines steady-state analysis and simulation, we call it the *hybrid simulation* approach.

This work has the following contributions: (1) Novelty: It develops a solution approach combining queueing models and simulation to capture the detailed streaming process of packets in the physical layer of a cloud-based service system to deduce the personal experience of customers under a given parameter setting of the system. (2) Precision: The exact analysis of the proposed model gives closed-form expressions of performance measures, on server side (i.e., the mean number of VMs required, the mean life cycle time of a VM (i.e., the mean time for a VM from being activated to joining the idle pool of VMs), and the number of customers at a VM) and on customer side (i.e., mean number of interruptions and mean lag time on a customer session). (3) Simulation: The novel simulation approach requires simulating the experience of individual customers on a single VM, not a complicated system with comings and goings of VMs and customers. (4) Application: As the solution approach can predict the service level for a given parameter setting of a cloud-based service system, it is an useful tool to project the amount of resource required in the design phase of the system.

The rest of the paper is organized as follows. Section 2 briefly reviews the literature on service quality of online streaming services. Section 3 is on the macro view of the system where the VMs interact with the customers. The section describes the service framework for cloud-based streaming services, the associated queueing models, the theoretical derivation, and the closed-form solutions of operation characteristics. Section 4 is on the micro view where the physical layer of the cloud-based service system interacts with customers. It provides simulation procedures and results on service quality analysis from the perspective of customers. Section 5 concludes the work.

2. Literature review

Taking advantage of a broadband connection and large server clusters, the entertainment and media industries have provided an excellent example cloud-based video services on the World Wide Web (WWW). For instance, Google retains global online video dominance with over 40% of videos viewed via YouTube worldwide. Furthermore, aimed at target audiences, websites or specific internet channels are beginning to mimic traditional TV programs to offer video programs (intertwining with advertisement), e.g., the service provided by Blinkx [2], the UK-based video search and advertisement network company that has acquired the US-based advertisement network company Burst Media in 2011. As a result of providing customers video-on-demand viewing experience, the online video service already makes up the largest component of global internet traffic (Budde Com [3]). There are similar surges of cloud-based services in other sectors, e.g., cloud gaming services, such as OnLive [34], StreamMyGame [43], and G-Cluster [19].

The rise of cloud-based streaming services demands a thorough analysis of such communication systems. In general the analysis of communication systems can be from two perspectives, of information system and of operations analysis.

For the information system perspective, empirical research is conducted to trace packet flows among servers and clients. Chen et al. [8,9] propose an empirical study on evaluating the quality and measuring the latency of cloud gaming services, showing that OnLive [34] has better performance than StreamMyGame [43]. For the content distribution networks of Google, Krishnan et al. [28] find out that even geographically near clients served by the same server can have round trip times differ for more than 50 ms, and empirically it is suboptimal to direct every client to the server of the least latency. Adhikari et al. [1] discover from the packet flow data that YouTube employs a proportional load balancing strategy among its seven data centers, and internet service providers can formulate strategy to response after understanding this YouTube's strategy and estimating the "unseen" traffic. There are studies on the transmission technology. Yin et al. [51] propose a hybrid approach that combines content delivery networks with per-to-peer (P2P) systems, achieving efficient delivery of live video streaming. Desmet et al. [17] compare P2P with the traditional client/server technology. They find that P2P is scalable and cost effective. Servers, clients, and VMs act like grids and the processing on packets, such as coding, transcoding, and decoding, act like jobs at grids. The packet flows through a communication network thus then lead to scheduling jobs and workflows in grids in a distributed environment. See the multi-criterion grid workflow scheduling in Wieczorek et al. [48], the advance reservation in Desmet et al. [17] to avoid congestion, and the dynamic resources allocation to minimize latency and cost in Ishii and Suzumura [25]. Bouten et al. [7] propose to adopt HTTP adaptive streaming and to take into account the in-network decisions during the rate adaptation process. They deploy optimization agents that monitor the available throughput using sampling-based measurement techniques and autonomous decision, based on the current buffer filling and network conditions, which quality representation it will download, keeping the ability to react to sudden bandwidth fluctuations in the local network. Krishnamoorthi et al. [29] leverage machine learning techniques for predicting buffer conditions on the client side via classification of HTTP and HTTPS traffic. We present a queuing model for cloud-based streaming media services that takes packet-level dynamics into account so that mixed-mode simulation can be used to

estimate customer-influenced performance. We derive the simulation program based on the analysis model, showing how the quality of service is affected by server and client performance.

There are also various heuristics to balance the load of machines and networks (Cherkasova [11], Pirkowski et al. [35], Wee and Liu [47], and Latre et al. [32], and an effective architecture for reliable large-scale stream computation in the cloud (Qian et al. [36]).

The analysis of operation perspective often appear in term of queueing models. Queueing models have been applied to analyze communication systems (e.g., Kleinrock [27] for some classical results, and Balsamo et al. [5] for more recent finite capacity models), computer performances (e.g., Riska et al. [38]), and the optimal number of servers in a server/clint environment (Son and Kim [41]). There are also queueing models on cloud services. Tewari et al. [44] compare two clustered architectures for servers providing video service. With queueing models for servers and disks, the paper studies the optimal storage clunk size of videos and the storage-ahead buffer size. Kang et al. [26] trace traffic in Yahoo! Video. Among other issues, they estimate the number of VMs needed for exponential and Pareto inter-arrival times.

As discussed in Introduction, we would like to study the customer experience and estimate the amount of resource for a given set of parameter values for online streaming service. The empirical studies and the various load balancing heuristics in the information system perspective give no clue on this. There are analytical models in both perspectives that estimate certain waiting time; however, in three aspects all of them miss our requirements.

First, all these waiting times are the *experience of individual packets, not of customers*. The waiting time for a customer, i.e., in our terminology the lag time of a customer, is *the time that the customer waits for the cloud service,* not the waiting of any packets. Second, more precisely, the lag time of a customer at his client (i.e., local device) is determined by the internet traffic, the arrival sequence of packets, and the capability of his client. The model must explicitly incorporate such features. Third, at the server side the production delay is partially determined by the workload of the virtual machine that serves the customer. Even under stationary demand pattern, the actual number of the clients serving by a virtual machine changes with time. This feature must be part of the model. The queueing analysis of the *round-robin* packet scheduling policy in computer systems captures the effect of multiple clients on a server, or a virtual machine in our context (Sakata et al. [39], and Rasch [37], Silberschatz et al. [40]). However, these models have no linkage with the internet traffic and the detailed dynamics of the physical layer. In this paper we develop a solution approach that incorporates all the three missing features in the literature. The idea of the presented approach has been proposed in an earlier work on quantitative analysis of cloud-based streaming services [52].

Finally, cloud simulators provide developers a systematic way to study cloud services, e.g., CloudSim in Calheiros et al. [13] and SimGrid in Casanova et al. [12]. It is known that simulation models that capture operations in detail are time consuming and intractable for large-scale systems. As shown in Velho et al. [46], even flow-level network models that rely on tractable mathematical manipulation may not fully capture the performance of cloud systems. While based on different model assumptions from those in Calheiros et al. [13], Casanova et al. [12], and Velho et al. [46], we share the same objective of providing an effective simulation approach for large-scale cloud systems. Our approach is capable to do so because it deduces the performance of the whole system by observing sample paths of individual customers on a single VM.

3. The dynamics of the cloud-based streaming service system

In a cloud-based streaming service system, the service is provided by a cluster of VMs that are dynamically allocated. The operation rules for VMs dictate the performance of the system. A common set of operation rules is to designate a fixed amount of VMs for a given cloud-based service. Then workloads are directed to VMs by balancing their workloads. Such a set of operation rules assigns customers to VMs with less workloads, which improves the service levels of the system in some sense. However, as discussed in Section 1, such a system is analytically intractable. The ideal amount of VMs required for a given demand pattern can only be deduced by trial-and-error through simulating large, complicated systems with statistically imprecise results. The operation rules proposed in the fourth assumption of Section 1 leads to a clean hybrid simulation approach that combines theoretical analysis with the simulation of the experience of *individual* customers on a *single* VM. The approach predicts the mean amount of VMs required for a given demand pattern, without locking up a fixed amount of VMs; the cloud-based service provider thus has greater flexibility in using its resources.

By the fourth assumption, a VM can be in three states, *working* (after initiation), *retiring* (after the number of customers reaches the upper bound B, and *idling* (after all the B customers at retirement have finished their sessions). At any time there is only one corresponding worker receiving incoming customers; however, at any time there can be multiple working VMs. The retiring VMs are in fact working; each retiring VM still serves existing customers till the disconnection of the last session.

The customer arrivals follow a Poisson process of rate λ. Upon arrival, (the request of) a customer takes a short duration δ time units before it joins the queue of the VM that serves as the corresponding worker. All VMs, being the corresponding worker or retiring, adopt the round-robin policy to process the customers waiting in queue (Rasch 1970, Silberschatz et al. 2004): The processing of any VM takes place in blocks of time, each block being referred to as a *quantum* Δ, of duration, e.g., from 0.01 to 100 ms. When his turn of service comes, a customer gets his quantum of service. Upon completion of the quantum, the customer leaves the system if his connection time is completed within the quantum of the serving VM else the customer re-joins the end of the queue of the VM, and the VM continues with the next customer in queue. A customer gets continuous service from a VM only if it is the only customer in the VM's queue. This round-robin policy is one common time-division multiplexing scheme.

Let $\mathscr{C}$ denote the generic *connection time* of a customer, which is the duration that a customer willing to spend on the cloud-based service. Such connection time of a customer includes his lag time waiting for the cloud-based service at an interruption. By the second assumption in Section 1 the connection times of customers, denoted as $\mathscr{C}_i$ for customer i, are independent and identically distributed exponential random variables of rate μ.

The operations characteristics of the system and the service levels of customers are determined by two controllable parameters, the maximal number B of customers simultaneously served by a VM and the duration of time quantum Δ that a VM allocates to each task in queue periodically, and the system characteristics, namely, the arrival rate λ, the service rate μ of connection times, and the operation rules adopted for VMs.

In this paper, we develop a hybrid simulation approach to estimate the service levels for customers for a given set of parameters B, Δ, λ, F. The approach is of two levels, the macroscopic level that considers the interaction of VMs and customers, and the microscopic level that considers the interaction of the physical layers of the cloud-based service system and customers. This section deals with the macroscopic level and the next for the microscopic level.

To analyze the system, it suffices to focus on VMs and ignore both the dispatcher and the monitor, i.e., effective δ is set to zero. The time taken for a dispatcher to process a customer is on average much shorter than that for a VM, and in fact be much shorter than the inter-arrival times of customers. Consequently, the system behaves as if the external arrivals directly arrive at the VMs. Similarly, there is enough resource assigned to the monitor such that the monitor does not have any effect

on experience of customers.

For rest of this section, for completeness, Section 3.1 gives the continuous-time Markov chain model of the system, which has no close-form solution. Section 3.2 gives the stationary distribution of the corresponding worker, a key result later to derive the operation characteristics and service levels of the system. Sections 3.3 and 3.4 give, respectively, two operation characteristics of the system, the mean life cycle time of a VM and the mean number of VMs required in the system.

3.1. The transition for exponential connection times

Let $\mathbf{s} = (n; n_1, ..., n_B)$ be the state of the system, where n is the number of customers at the corresponding worker, $n \le B-1$; n_j is the number of retiring workers with j customers at the machine, $j \in \{1.B\}$.

Further define notation $\mathbf{s}_+ = (n+1; n_1, ..., n_B)$ to be the state with one more customer at the corresponding worker than $\mathbf{s}$; $\mathbf{s}_- = (n-1; n_1, ..., n_B)$ to be the state with one less customer at the corresponding worker than $\mathbf{s}$; $\mathbf{s}_R = (0; n_1, ..., n_B + 1)$ to be the state with no customer at the corresponding worker and one more retiring worker with B customers than $\mathbf{s}$; $\mathbf{s}_{j-} = (n; n_1, ..., n_{j-2}, n_{j-1}+1, n_j\text{-}1, n_{j+1}, \cdots, n_B)$ to be the state with one more (j-1)-customer retiring worker and one less j-customer retiring worker than $\mathbf{s}$, $j > 1$; $\mathbf{s}_{1-} = (n; n_1\text{-}1, n_2, ..., n_B)$ to be the state with one less 1-customer retiring workers than $\mathbf{s}$.

Note that the connection times are independent of how the customers are served. Thus, if two customers are with a VM at any time, the instantaneous rate of the departure of customers from the VM is of $2 \times \mu$, not μ as a standard processor-sharing system does. Then the infinitesimal rates of state $\mathbf{s}$:

- Total rate out: $q_{\mathbf{s}} = \left[\lambda + \left(n + \sum_{j=1}^{B} j \times n_j\right) \times \mu\right]$;
- Effect of an arrival:
  - $q_{\mathbf{s},\mathbf{s}_+} = \lambda$, for $n \le B$-2;
  - $q_{\mathbf{s},\mathbf{s}_R} = \lambda$, for $n = B$-1;
- Effect of a departure:
  - $q_{\mathbf{s},\mathbf{s}_-} = n \times \mu$
  - $q_{\mathbf{s},\mathbf{s}_{j-}} = n_j \times \mu \times j$, for $n_j \ge 1, \forall\, j \in \{1, ..., B\}$.

Conceptually we have completely specified the infinitesimal rates of the continuous-time Markov chain (CTMC) that models the number of customers at the various VMs. It can be shown that the CTMC is positive and irreducible, and its stationary distribution can be found. The performance measures such as the mean number of VMs required, etc., follow accordingly. The only problem is that the state space is too large to make numerical computation realistic. In the following we develop an approach in which it suffices to analyze a single VM, the corresponding worker.

3.2. Stationary distribution of the state of the corresponding worker

To find the stationary distribution of the corresponding worker, let the state be the number of customers served by the corresponding worker. Fig. 1 shows the transition diagram.

At initiation, a corresponding worker serves no customer and is at state 0. As customers come and go, at some point the corresponding worker is at state $B-1$ and a customer arrives. Recall that a corresponding worker starts to retire if there are B customers. Thus the new arrival triggers the retirement of the corresponding worker. The current corresponding worker goes retiring and a VM is initiated to be the *new* corresponding worker. The transition from state $B-1$ to state 0 actually models the simultaneous retirement of the current and the initiation of the new corresponding worker.

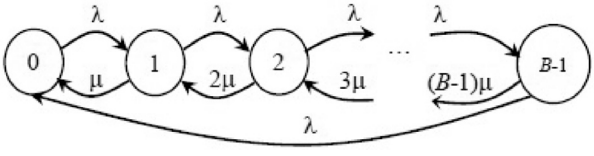

Fig. 1. The transition diagram for the number of customers at the corresponding worker.

Let p_i denote the probability of state i, $i \in \{0, ..., B-1\}$. We have the following set of balanced equations:

$$\begin{cases} \lambda \times p_B + \mu \times p_1 = \lambda \times p_0, \\ \lambda \times p_{i-1} + (i+1) \times \mu \times p_{i+1} = \\ (\lambda + i \times \mu) \times p_i, \forall\, i \in \{1, ..., B-2\}, \\ \lambda \times p_{B-2} = (\lambda + \mu \times (B-1)) \times p_{B-1}. \end{cases} \tag{1}$$

Let $\xi = \lambda/\mu$ be the ratio between the arrival rate λ and the disconnection rate μ of customers; the parameter ξ is an indication of how busy the system which is the traffic intensity in a standard $M/M/1$ queue. We say a system is *lazy* when $\xi < 1$, and a system is *busy* when $\xi \ge 1$. Solving Eq. (1), for $k \in \{2, ..., B\}$, we have:

$$p_{B-k} = \left[\sum_{m=1}^{k}\left(\prod_{j=m}^{k-1}\left(\frac{B-j}{\xi^{k-m}}\right)\right)\right] \times p_{B-1}, \tag{2}$$

where by convention $\prod_{j=k_1}^{k_2}(\cdot) = 1$ when $k_1 > k_2$. Applying the normalization equation to Eq. (2), we have, for $k \in \{1, ..., B\}$, the closed form solution:

$$p_{B-k} = \frac{\sum_{m=1}^{k}\left(\prod_{j=m}^{k-1}\left(\frac{B-j}{\xi^{k-m}}\right)\right)}{1 + \sum_{l=2}^{B}\left[\sum_{m=1}^{l}\left(\prod_{j=m}^{l-1}\left(\frac{B-j}{\xi^{l-m}}\right)\right)\right]}. \tag{3}$$

Given λ, μ, and B, one can compute via Eq. (3) the distribution of customers in the corresponding worker, which in turn gives L, the average number of customers in a typical corresponding worker.

$$L = \sum_{k=1}^{B-1} p_k \times k. \tag{4}$$

The two parameters ξ and B determine the value of L. To get some insights on how L is affected by ξ and B, we plot in Fig. 2 the value of L for a busy system with lots of customers as ξ changes from 1 to 100, and in Fig. 3 for a lazy system with arrival rate no more than service rate as ξ changes from 0.01 to 1. In both figures graphs for B equal to 5, 10, 20, and 50 are shown.

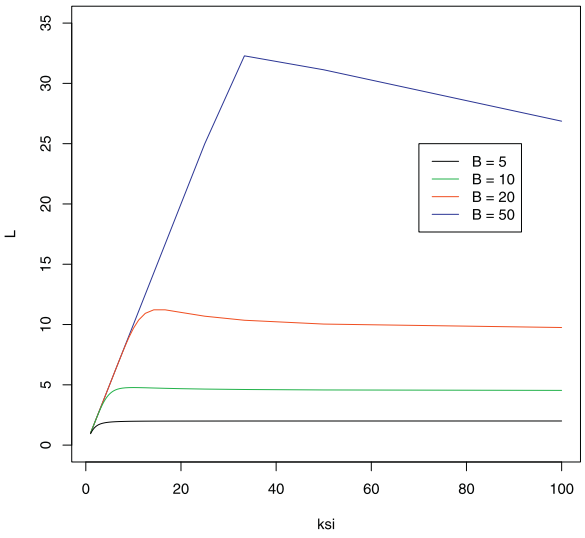

Fig. 2. Numeration on L for $\xi \ge 1$ (a busy system).

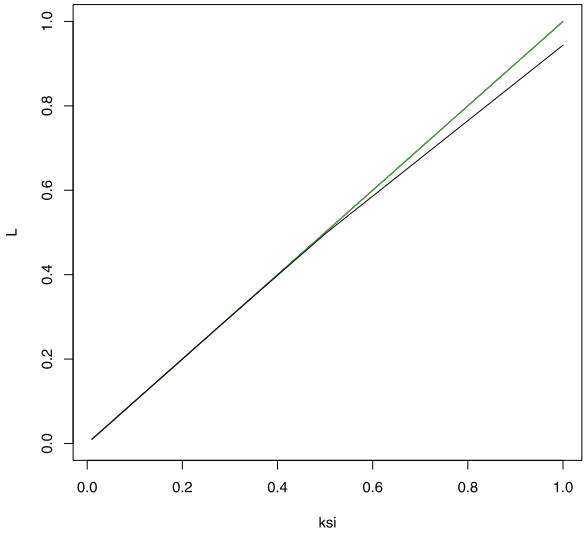

Fig. 3. Numeration on L for $\xi \leq 1$ (a lazy system).

The main objective of simulation design is to cover a reasonable range of arrival/service ratios under sampled parameter settings. It is essential that we can observe the turning point of customer lengths on average when customer status changes. The result of the ratio (arrival/service rate) set from 1 to 100 is reported in the paper. Figs. 2 and 3 show that ratios out of this range (more lazy or more busy) more likely fall in the two tails that have relatively stable and less L in the queuing model (or number of interruptions and lag time as later shown in the simulation result).

There are three important observations.

- *Observation 1: L appears to be a mono-peak function of ξ for a busy system.*
- *Observation 2: The ξ value of the peak of L increases when B increases.*
- *Observation 3: L appears to approach ξ for a lazy system.*

In general L first increases and then decreases as ξ increases (i.e., as λ increases for a fixed μ, or as μ decreases for a fixed λ). At first sight such behavior looks odd: In conventional queueing models in general the mean number of customers increases with the traffic intensity. Recall that the transition diagram in Fig. 3.1 is *not* for a conventional system. As the traffic intensity increases, it is more likely for an arriving customer finds the corresponding worker at state $B-1$, and changes the state to 0 for initiating a new corresponding worker. This leads to L being a mono-peak function of ξ, where we explain by increasing λ and holding μ fixed; a similar argument goes through by changing μ and holding λ fixed: When λ is much less than μ, the quick service rate pulls the state to stay in small states. As λ increases (i.e., ξ becomes larger), more arriving customers drag the system to larger states. Thus, in this range L increases as ξ increases. As ξ increases further beyond certain threshold, the value of λ relative to μ has become so large that the corresponding worker reaches state B easily and turns the state to 0. The value of L decreases as ξ increases in this range.

In short, the result of Observation 2 is the tradeoff between two driving forces to reduce L, by reaching the upper bound to invoke a new corresponding worker, or to remain at states with small number of customers by having customers leaving more frequently. For larger B, it is more difficult to reach the upper bound, and it takes smaller μ for a fixed λ (i.e., a larger ξ, a busier system) to reach the upper bound. Thus, it takes a larger ξ (i.e., a busier system) for the peak to occur as B increases.

In the following, we are going to give analytical explanation of the shape of L. Consider first a busy system such every hour there are 900 requests, and on average each customers stay for 20 minutes. Then $\lambda = 15$ per minute, $\mu = 0.05$ per minute, and $\xi = 300$. For ξ of this magnitude and $B = 50$, $B/\xi < 1$. From Eq. (2), ignoring terms of $1/\xi^2$ or smaller, $p_k \approx (k+1)p_{B-1}/\xi$ for $k \in \{0, ..., B-2\}$. Normalizing and solving, $p_{B-1} = 1/\left[B + \frac{B(B-1)}{2\xi}\right]$, $p_k = \left[1 + \frac{(k+1)}{\xi}\right]/\left[B + \frac{B(B-1)}{2\xi}\right]$, from which we have

Lemma 1. *For $\xi \gg 1$, $L \approx \frac{2(B+1)}{3} - \frac{B(B+7)}{6\left[B + \frac{B(B-1)}{2\xi}\right]} \approx \frac{B-1}{2}$.*

With this approximation, for $L = 2.0132$, 4.5419, 9.6381, and 25.2173 for $B = 5$, 10, 20, and 50, respectively. Check from Fig. 2 that the approximation is accurate for small value of B in which $\xi \gg 1$. The result is not too off from the true value even for $B = 50$. Now for a more relaxing system, we have

Lemma 2. *For $\xi \ll 1$, $L \approx \xi$.*

Fig. 3 shows the numerical result that is consistent with Lemma 2, whose analytical justification is the following: When $\xi = 1$, the last terms inside the pair of square brackets of Eq. (3)for p_0, p_1, and p_k, $k \geq 2$, are $(B-1)!/\xi^{B-1}$, $(B-1)!/\xi^{B-2}$, and $\frac{(B-1)!}{k!\xi^{B-k-1}}$, respectively. These last terms dominate other terms. When $\xi \ll 1$, the effect of these last terms is even more dominating than when $\xi = 1$. Thus, we have the pattern of $p_0 \approx p_1/\xi \approx k!p_k/\xi^k$, for $k \geq 2$. The normalization equation $\sum_{k=0}^{\infty} p_k = 1$ implies that $p_0 = e^{-\xi}$, and thus, we have $L \approx \sum_{k=0}^{\infty} k \times p_k = \xi$. It does not take a large B to have an accurate approximation; e.g., for $\xi = 0.01$ with $B=5$, $\frac{1}{\sum_{k=0}^{4} \frac{\xi^k}{k!}} - e^{-\xi} = 8.18012 \times 10^{-13}$. The analytical justification shows that as B increases for $\xi < 1$, p_0 becomes larger, which is also consistent with the numerical results.

3.3. The mean life cycle time of a virtual machine

A VM goes through the process of being activated, followed by retirement, and then eventually returning to the idle pool. Call such a process the *life cycle* of a VM. We are going to deduce the mean cycle time. As shown in the next two subsections, such information is essential to deduce the performance measures for the system.

Let N_w be the number of customers at a VM when it is working. Refer to Fig. 4 for the transition diagram for N_w, where $N_w = j \in \{0, ..., B-1\}$ when the VM is the corresponding worker serving j customers, and $N_w = j_r \in \{0_r, 1_r, ..., B_r\}$ when it has retired and still with j_r customers. A VM is first activated to state 0, retires at state B_r, and then becomes idle at state 0_r.

Let T_{0,B_r} be the time taken to go from state 0 to state B_r and $T_{B_r,0_r}$ be the time taken to go from state B_r to state 0_r. The cycle time $T_{0,0_r} = T_{0,B_r} + T_{B_r,0_r}$. It is straightforward to have:

$$E\left(T_{B_r,0_r}\right) = \sum_{k=1}^{B} \frac{1}{k \times \mu}. \tag{5}$$

To find out $E(T_{0,B_r})$, let T_{k,B_r} be the first-passage time from state k to state B_r. For $k \in \{1, ..., B\}$,

$$T_{B-k,B} = \tau_{B-k} + \begin{cases} T_{B-k+1,B}, & w.\ p. \quad \dfrac{\lambda}{\lambda + (B-k) \times \mu}, \\ T_{B-k-1,B}, & w.\ p. \quad \dfrac{(B-k) \times \mu}{\lambda + (B-k) \times \mu}, \end{cases} \tag{6}$$

where τ_{B-k} is the sojourn time in state $B-k$. Let $\gamma_j = E(T_{j,B_r})$. Let the notation $(\cdot)^T$ be the transpose of $(\cdot)$. Define

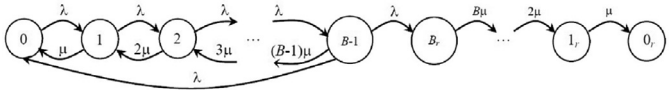

Fig. 4. The transition diagram for the number of customers with a worker.

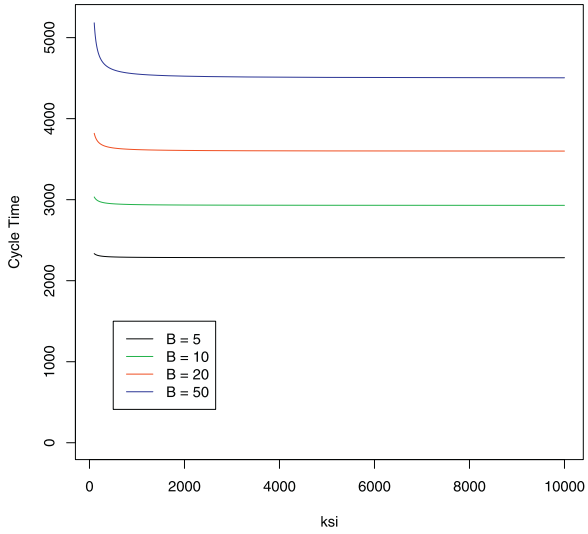

Fig. 5. The mean cycle time (in s) of a worker ($\xi \geq 100$, $\mu = 0.001$).

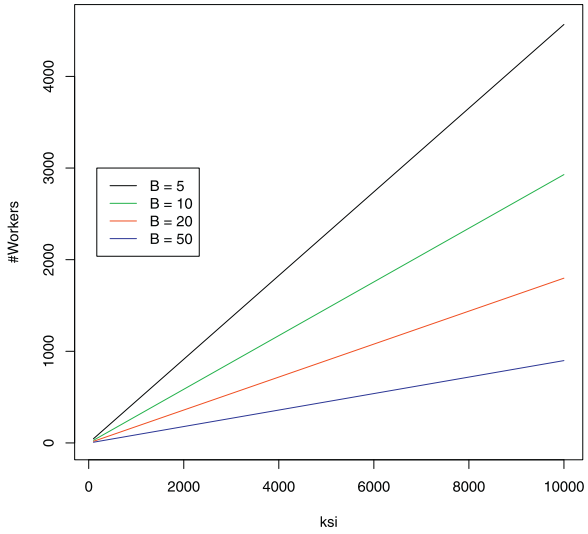

Fig. 6. Number of workers in system ($\xi \geq 100$, $\mu = 0.001$).

$$\Gamma \equiv [\gamma_0, \gamma_1, ..., \gamma_{B-1}]^T,$$

$$\Delta \equiv \left[\frac{1}{\lambda}, \frac{1}{\lambda_\mu}, \frac{1}{\lambda + 2 \times \mu}, ..., \frac{1}{\lambda + (B-1) \times \mu}\right]^T,$$

$$\Lambda \equiv [a_{ij}]_{0 \leq i,j \leq B-1},$$

with $a_{j,j} = 1, j = 0, ..., B$, $a_{01} = -1$, $a_{j,j-1} = -\frac{j \times \mu}{\lambda + j \times \mu}, j = 1, ..., B$, and $a_{j,j+1} = -\frac{\lambda}{\lambda + j \times \mu}, j = 1, ..., B-2$. We have the following theorems:

Theorem 1. *Λ is invertible.*

Theorem 2. *$\Gamma \equiv \Lambda^{-1}\Delta$, and $E(T_{0,B_r}) = \gamma_0$.*

Fig. 5 plots the mean cycle time of a VM against ξ and B for a fixed μ. The results are intuitive: For a given B, the mean cycle time of a VM decreases as ξ increases, i.e., everything else the same, a less busy VM has shorter cycle time. As ξ increases further, the arrival rate is so high that an initiated VM reaches retirement in very short time, and the cycle time is basically the retirement time of the VM. For a given ξ, the mean cycle time of a VM increases with B, i.e., everything else the same, a VM with larger B has longer cycle time. For any B, as ξ increases, the mean cycle time converge to B-dependent constant. This intuition is that the mean cycle time $E(T_{0,0_r}) = E(T_{0,B_r}) + E(T_{B_r,0_r})$, where for a fixed μ, $E(T_{0,B_r})$ decreases as λ increases while $E(T_{B_r,0_r})$ remains constant.

3.4. The mean number of working virtual machines in system

Within one cycle time, on average a VM works for $E(T_{0,0_r}) = E(T_{0,B_r}) + E(T_{B_r,0_r})$ units of time, and on average accepts $\lambda \times E(T_{0,B_r})$ arrivals. Because the rate of customers handled by one VM is

$$\frac{\lambda \times E(T_{0,B_r})}{E(T_{0,B_r}) + E(T_{B_r,0_r})},$$

on average there are

$$1 + \frac{E(T_{B_r,0_r})}{E(T_{0,B_r})} = 1 + \frac{\sum_{k=1}^{B} \frac{1}{k \times \mu}}{E(T_{0,B_r})} \tag{7}$$

working VMs employed by the system. This average number of working VMs required serves as one of the performance indicators of the system. Note that $E(T_{B_r,0_r})$ can be significantly larger than $E(T_{0,B_r})$; e.g., when $\lambda \gg \mu$, many VMs are required to substantiate the system.

Fig. 6 plots the mean of number of working VMs in a system against ξ and B. The number of working VMs appears to be linear in ξ for a given B, where for the same ξ, a larger value of B corresponds to a smaller mean number of working VMs. This observation is consistent with (7): When $\lambda \gg \mu$, $E(T_{0,B}) \approx \frac{B}{\lambda}$ so that the mean number of working VMs in system is linear in ξ.

As shown in Fig. 6, a large number of working VMs is required for a busy system with small B. In those cases it is necessary to prepare huge resources for coping with high demands. Fortunately, for any particular application peak demands only occur sparsely, and often at different time among applications. One can take advantage of resource sharing schemes among applications to avoid excessive resource for the cloud-based platform.

4. Microscopic model: Model for the service experience of customers

In this section, we model the detailed dynamics of the physical layer of the cloud-based service system, and its consequences on the service levels of customers. We trace how a VM processes the packets for its customers.

As defined, each VM adopts the *round-robin* rule to serve its assigned customers, each round for quantum time units Δ on a customer. When there are n customers with a VM ($n \leq B$), the VM serves all n customers via regularly sending each customer a set of packets in one out of $n \times \Delta$ time units, in the order of their positions in the queue of the VM.

Consider a particular customer in the subsequent discussion. By assumption, the ith set of packets of the customer, i.e., the set of packets generated in the ith time quantum for the customer by the VM, takes $T(i) + D(i)$ time units to reach the customer, where $T(i)$ is a customer dependent transmission time function (e.g., dependent on the location of the customer) and $D(i) \sim i.i.d.$ random variables of a given distribution to capture the random web traffic. It also has been shown empirically that $T(i) + D(i)$ are independent and identically distributed, and follow a heavy tailed distribution [14] for the same customer at the same location with the same kind of packets.

The ith set of packets takes the duration $S(i)$ to be processed at the end device of the customer, where $S(i)$ can be a device-dependent constant. Naturally, if everything else is held fixed, the chance for a customer to experience time lag increases with the runtime value of n of a worker. In this section, we discuss a procedure to estimate the mean number of interruptions and mean total time lag of a customer given the set of parameters λ, μ and B, Δ decision variables. To do so we need to understand the dynamics of packet processing at a customer's device.

Let $A(i)$ be the arrival epoch of the ith set of packets at the customer; $E(i)$ be the epoch that the ith set of packets has completely been decoded and starts to take effect at the customer. In general, $A(i)$ may not

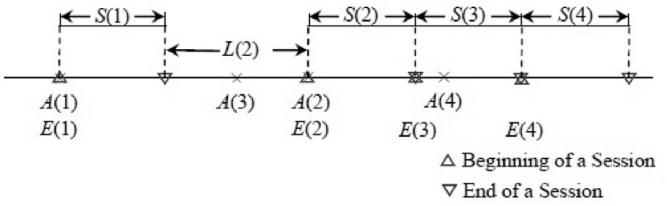

Fig. 7. The relationship among $A(\cdot)$, $E(\cdot)$, and $S(\cdot)$.

be equal to $E(i)$, and there is a *lag time* $L(i)$ for the i^{th} set of packets if $E(i-1)+S(i-1) < E(i)$. See Eqs. (8) and (10) for the derivation of $\{E(i)\}$, $\{L(i)\}$ from $\{T(i)\}$, $\{D(i)\}$. Fig. 7 provides intuition for the subsequent derivation. In this example, the end device is idle from $A(1)+S(1)$ to $A(2)$ during which the customer is also idle there waiting for the arrival of the second set of packets. Thus, $L(2) > 0$. As shown in Fig. 7, $L(1) = L(3) = L(4) = 0$. If $A(5) > E(4) + S(4)$, the customer's end device queue empties at $E(4) + S(4)$ and the customer needs to wait for the arrival of the 5th set of packets to resume service.

Suppose that a VM starts working for a customer at epoch I. At $I + \Delta$, the worker has finished processing the first quantum time units for the customer, and at $A(1) = I + \Delta + T(1) + D(1)$ the set of packets arrives at the customer's end device. On average packets accumulate at a customer's end device as time goes on. To avoid an interruption at the early stage of service, many streaming service designs [18] starting displaying the first set of packets of a customer only after the accumulation of N_{init} sets of decoded packets at the customer's end device, where N_{init} is a pre-specified constant. Consequently, $E(1)$ the effect time of the first set of packets can be taken as the time to accumulate the first N_{init} sets of packets for the customer.

Let n_j be the number of customers that the VM serves within the period between the generation of the $(j-1)^{st}$ and jth set of packets. n_j changes with the arrivals and departures of customers. In the simulation procedure described in Section 4.1, along the simulated sample path of a tapped customer, n_j are known quantities.

In general,

$$A(i) = I + \sum_{j=1}^{i-1} (n_j \times \Delta) + \Delta + T(i) + D(i), \forall\, i \geq 1; \tag{8}$$

$$E(1) = \text{MAX}\{A(i), \forall\, 1 \leq i \leq N_{init}\}. \tag{9}$$

$$E(i) = \text{MAX}\{E(i-1) + S(i-1), A(i)\}, \forall\, i \geq 2. \tag{10}$$

Take $L(1) = E(1) - A(1)$.

$$L(i) = [A(i) - (E(i-1) + S(i-1)]^{+}, \forall\, i \geq 2. \tag{11}$$

$L(1)$ is the initial delay spent on accumulating N_{init} sets of packets without showing to the customer. The lag time experienced by the customer can be accumulated from Eq. (11). Let $\mathscr{C}$ be the connection time that a customer spends on the cloud-based service, $\mathscr{C} \sim F$; and Ω be the number of sets of packets processed by the local device within $\mathscr{C}$.

$$\mathscr{C} = \sum_{i=1}^{\Omega} (L(i) + S(i)). \tag{12}$$

4.1. The simulation of the physical layer

We develop a simulation procedure to deduce the customer experience by capturing the dynamics in the physical layer of the cloud-based service system. In particular, it makes use of results developed in Section 3.2 to construct a hybrid simulation procedure that simulates the processing of tapped customers packet by packet at the quantum level of a *single* VM, not a full system, along the sample paths of the tapped customers. The procedure deduces the service quality of the system for various system parameter settings of B and ξ.

Suppose we keep on simulating the life cycle of one VM, from being initiated as the corresponding worker, to retiring as the number of customers reaching B, and finally joining the pool of idle VMs after the departure of the last customer on hand. Then VM accepts new customers when it is the corresponding worker. Each arriving customer comes with its own connection time, which is a known random variate from exp(μ), the distribution of the connection time of the customer. An arriving customer is called a *k-customer* if on arrival the customer finds k customers being served at the corresponding worker. Let us trace the sample paths of *tapped* k-customers from their arrivals to departures, $k \in \{0, ..., B-1\}$. The state transitions in Fig. 3.1 dictates the state changes faced by such customers. Between any two state changes, the number of customers n at the corresponding worker is known. Then the set of Eqs. (8) to (13) can deduce the arrivals of packets to the tapped customers' local devices, and hence deduce the number of interruptions and the lag times encountered by the tapped customers. By repeating tapping the sample paths of k-customers multiple times, we can get the average number of interruptions and average lag time along the sample path of k-customers. The process can be done for all $k \in \{0, ..., B-1\}$.

Note that the arrivals follow a Poisson process. By Poisson Arrivals See Time Averages (Wolff [49]), the probability of an arrival finding k existing customers at the corresponding worker is exactly the stationary distribution found in Eq. (3). Weighing the average number interruptions and average lag times of k-customers with respect to this stationary distribution, we have the mean number of interruptions and mean lag time in a connection session of customers.

We break the simulation process into three algorithms below. Algorithm 1 describes the overall simulation shell to collect statistics from NR replications of k-customers. The algorithm sets parameter values and initializes variables, simulates the state changes of the end device induced by arrivals and departures, and eventually collects the average lag time L_k and average number of interruptions I_k of the k-customers. Algorithm 2 gives the detailed simulation on a single event, embedding in which Algorithm 3 is invoked to compute the lag times and interruptions according to equations Eqs. (8),(10), and(11) between two state changes of the end device. See Table 1 for the parameters used in the three algorithms.

In Algorithm 1, the priority event queue E is the event calendar in a typical discrete-event simulation system, i.e., it arranges events in ascending order of their event times, and the earliest event is taken out to execute from E at an event time *e.time*. The detailed tasks to execute at an event depends on the event type *e.type*, `end` for the termination of simulation, `arrival` for an arrival of a new customer, and `departure` for the end of connection time of an existing customer. Some task includes the creation of a future event. In that case, the event type *e.type* and event time *e.time* are generated. The event is *enqueued* (i.e., inserted) to E if it occurs before $\mathscr{C}$, the departure of the tapped customer. When the flag f indicates that the corresponding worker has retired, no more `arrival` event will be created.

Step **1** of Algorithm 1 sets the values of *NI, LT, NR*, and *m*. An iteration of the while loop in step **2** simulates a replication of the procedure. Step **2.1** initialize current time *cur*, priority event queue E, state variable s (i.e., number of customers at the corresponding worker), and flag f. Step **2.2** sets the connection time $\mathscr{C}$ of the tapped customer, and thus create an `end` event in E. Step **2.3** sets the disconnection time of each k−customer on arrival, k= 1 to $B-1$. There is exactly one departure event for an arrival. Those departure events that happen within the connection time are added to the event queue E. Step **2.4** sets the time of an `arrival` event and inserts to E if necessary. Step **2.5** simulates events in E seriously according to *e.time* until E becomes empty. Step **2.6** completes one replication of simulation, and gets back to the loop of step **2** for the next replication. Step **3** outputs the number of interruptions (I_k) and the accumulated lag time (L_k).

Algorithm 2 consists of two main parts, first (Step **2**) to call Algorithm 3 to simulate the detailed lag times and interruptions between two state changes of the end device, and second (Step **3**) to

INPUT: k, μ (the service rate), λ (the arrival rate), Δ, B, the distribution of transmission time (i.e., $T(i) + D(i)$ for the ith set of packets), $S(i)$ the mapping function for the processing time of the ith set of packets on the client side.
OUTPUT: the experienced lag time (L_k) and # of interruptions (I_k).

1 Set NR; and set LT, NR, m to 0.

2 While ($m \leq NR$) {

2.1 Set cur to 0. Initialize an empty E ($e \in E$ is sorted by $e.time$). Set s to k+1; Set f to `true` if $s = B$; to `false`, o.w.

2.2 Draw a random variate t from exp(μ); Set C to t; create an `end` event e (i.e., set $e.type$ to `end` and $e.time$ to t, and enqueue e to E).

2.3 Create a departure event for each $k-$customer (i.e., for each event e, set $e.type$ to departure; draw a random variate t from exp(μ); if $t < C$, set $e.time$ to t and enqueue e to E; ignore the event e, o.w.)

2.4 If f = `false`, create an `arrival` event e (i.e., set $e.type$ to `arrival`. Draw a random variate t from exp(λ); If $t < C$, set $e.time$ to t and enqueue e to E).

2.5 While (E is not empty) call **Algorithm 2**;

2.6 Set m to m+1;
}

3 $I_k = NI/NR$ and $L_k = LT/NR$.

Algorithm 1. The simulation procedure for the end device queue for the k-customer, $k < B$.

1 Get the earliest event e from E.

2 Call **Algorithm 3** to simulate the end device with the worker stays at state s in (cur, $e.time$).

3 At $e.time$, switch ($e.type$) {

3.1 case `end`: break;

3.2 case `departure`: Set s to s-1; break;

3.3 case `arrival`: Set s to s+1. If f = `false` and $s = B$, set f = `true`. Create a departure event e' for e (i.e., set $e'.type$ to `departure`; draw a random variate t from exp(μ); if $t + e.time < C$, set $e'.time$ to $t + e.time$ and enqueue e' to E); if f = `false`, create the next arrival event e'' (i.e., set $e''.type$ to `arrival`; draw a random variate t from exp(λ); if $t + e.time < C$, set $e''.time$ to $t + e.time$ and enqueue e'' to E); break;

}

4 Set cur to $e.time$;

5 Return to Step 2.5 of Algorithm 1.

Algorithm 2. The simulation procedure on one event.

INPUT: s (number of customers served by the worker), t_1 (begin time), t_2 (end time)

1 Note the values of $A(i)$, $S(i)$, $E(i)$, cumulative number of interruptions, and cumulative lag time of the tapped k-customer from the last call to Algorithm 3.

2 Within the period (t_1, t_2), simulate the end device queue with s customers at the VM. For each Δ time units generate a set of packets. Send the sets of packets to the s customers in the round-robin fashion. For each set of packets, compute its arrival time and its effect time (according to Eqs. (8), (9) and (10). A set of packets is processed if its effect time is less than C. If there have been more than N_{init} packets, set $NI = NI+1$ whenever an interruption occurs for a set of packets, as well as accumulate the lag time of the customer to LT.

3 At t_2 mark the values of $A(i)$, $E(i)$, cumulative number of interruptions, and cumulative lag time of the tapped k-customer and return to Step 2 of Algorithm 2.

Algorithm 3. The simulation procedure on the end device, where the worker stays at state s in (t_1, t_2).

Table 1
The parameters for the simulation procedure.

Name	Definition
$\mathscr{C}$	The connection time of the customer, i.e., i.i.d. exp(μ).
NR	The number of replications of the simulation procedure.
NI	The cumulative number of interruptions within connection time $\mathscr{C}$ for *NR* replications.
LT	The cumulative experienced lag time within connection time $\mathscr{C}$ for *NR* replications.
e	An event *e* has two fields: *e.time* indicates the epoch that *e* happens, and *e. type* ∈ {end, departure, arrival} indicates that the event is termination, service completion, and arrival, respectively.
E	The set of events that have not happened yet, but are sorted according to *e.time* and stored in a priority queue.
I_k	The estimate of mean number of interruptions in $\mathscr{C}$ for the *k*-customer.
L_k	The estimate of mean lag time for the *k*-customer.
f	A boolean to indicate whether the worker is in the retiring stage.
cur	The time that the customer has been in the system.

simulate event change for the server. Step **1** of the algorithm gets the earliest event from *E*, whose event time *e.time* by construction is less than $\mathscr{C}$. Step **2** calls Algorithm 3 to simulate the delivery of packets and accumulates lag time and interruptions accordingly. Step **3** simulates event changes, redirecting the simulation flow to Step **4** for the `end` event (Step **3.1**), reducing *s* by 1 for a `departure` event (Step **3.2**), and carrying out the tasks for an `arrival` event (Step **3.3**). In the last case, *s* is increased by one; *f* is updated if necessary; the departure time generated for the arrival is inserted in *E* only if it occurs earlier than $\mathscr{C}$; if the worker has not yet retired, a future arrival is generated, and is inserted to *E* if its time to occur is earlier than $\mathscr{C}$. Step **4** ends the simulation of one event and updates *cur* to *e.time*. Step **5** returns the control to Step **2.5** of Algorithm 1 where the step is executed until *E* becomes empty.

Algorithm 3 takes two inputs: the number of customers served by the worker *s* and the inter-event duration (*cur, e.time*). Step **1** notes the values of the statistics left from the last call of Algorithm 3 for the tapped *k*-customer. Step **2** is the main simulation step that generates sets of packets within (*cur, e.time*), compute the arrival times and effect times of the sets of packages, and accumulate output statistics. Step **3** stores the values of the statistics at the end of Algorithm 3 and return the control to Step **2** of Algorithm 2.

There are subtleties to simulate the round-robin schedule of a VM. The inter-event duration is not necessarily in multiple of Δ, and the number of customers to be served by a VM before the first service of the tapped customer in (*cur, e.time*) generally changes with time and event. Such implementation details do not affect the simulation results and are skipped here for clarity.

With I_k and L_k found at the end of Algorithm 1 for $k = 0, ...,B - 1$: The (unconditional) mean total lag time in a connection session is:

$$\sum_{k=0}^{B-1} p_k \times L_k, \tag{13}$$

And, the (unconditional) mean number of interruptions in a connection session is:

$$\sum_{k=0}^{B-1} p_k \times I_k, \tag{14}$$

where p_k is the distribution given in Eq. (3).

4.2. Typical simulation results

It has been shown by Nossenson and Attiya [33] that the transmission times *X* for a particular file sending through Web for a particular server-client pair follow a *Pareto* distribution, therefore of heavy tailed, confirming the result shown in [14]. As the distribution function *F* and the density function *f* defined by Eq. (15), the Pareto distribution [23] has two parameters α and *k*, where the *shape* parameter α is responsible for the heavy-tail of the Pareto distribution and the *location* parameter *k* determines the lower bound of the possible value that a random Pareto variable can take on.

$$\begin{aligned} F(x \mid \alpha, k) &= 1 - \left(\frac{k}{x}\right)^{\alpha}; \\ f(x \mid \alpha, k) &= \frac{\alpha \times k^{\alpha}}{x^{\alpha+1}}; \\ k \le x < \infty; \alpha, k &> 0. \end{aligned} \tag{15}$$

It follows that:

$$E(X) = \frac{\alpha \times k}{\alpha - 1}, \alpha > 1. \tag{16}$$

While adopting the Pareto distribution to estimate transmission duration ($X_i = T(i) + D(i)$), we use the lower bound of $T(i)$ as the value of parameter *k* (the minimal transmission time) and estimate α based on Eq. (16) given an observed $E(X)$. There can be other ways to estimate the parameter values of a Pareto distributions [4] and [6].

We have implemented the simulation procedure in Algorithms 1 to 3. In our simulation runs, the duration of a quantum Δ is set to *QRate* × 0.0001 s, where the factor *QRate* to control the length of a quantum is set to be 1, 5, and 10 for comparison. The service rate μ is fixed at 0.005 (per second), while $\lambda = \mu/(0.01 \times i) = 1/2i$ for $i = 1$ to 20, i.e., λ increases non-linearly from 0.025 to 0.5 as ξ increases from 5 to 1000. The parameter *B* takes up values 5, 10, 20, and 50. The total transmission time, including the random delay, $T(i) + D(i) = X_i$, is a random variate from the Pareto distribution with $\alpha = 3$ and $k = 10 \times \Delta$. This construction dictates the minimum transmission time to be 10 times of Δ.

On the client side, for each set of packet *i*, $S(i) = 10 \times \Delta$ is defined as a linear function to Δ. It is assumed to be 10 times slower on the client side than on the server side to process packets so that normally the end device of a customer takes the amount of $10 \times \Delta$ time to process the set of packets produced by the worker in Δ. The number of initial sets of packets (N_{init}) that the end device accumulates before processing the first set of packets is set to 1, 100, 10,000 for comparison.

The number of replications (*NR*) is set to 1000. That is, for each setting (given *B* and ξ) we run the simulation procedure 1000 times for each *k*, find the average results from the 1,000 *k*-customers, $0 \le k < B$, and report the average values of the simulation results that are weighted with $\{p_k\}$ in Eq. (3). Consequently, a large number of simulation runs is conducted. For instance, for $B = 50$, the total replications in our simulation is equal to $50 \times 20 \times 1000$ for a given Δ(or *QRate*) and N_{init}.

The average experienced lag time and interruption by the customer in one sample replication.

For each pair of Δ (or *QRate*) and N_{init}, we run the simulation procedure and report its lagtime (accumulated from any lag time experienced) and interruption (increased by 1 every time for a lagtime greater than 0.001 s).

Figs. 8–10 plot *LagTime* (Eq. (13)) for ξ changes from 5 to 100, with Qrate set to 1, 5, and 10, respectively. Note that in these figures, the dotted lines show the initial delay during the accumulation of N_{init} sets of packets and the solid lines are the subsequent lag times accumulated from the following sets of packets.

The types of graphs are similar in shape to the graph of *L* (Fig. 2) and have a peak against ξ in a busy system. The quality of the system does not deteriorate without bound as ξ increases. With worst case quality at the peak values, the design of our system can cope with high demands for large ξ. The intuition is that for a busier system, the cloud platform by design has more workers employed to serve the coming/existing customers.

There are several factors that may affect the curves of lag time and

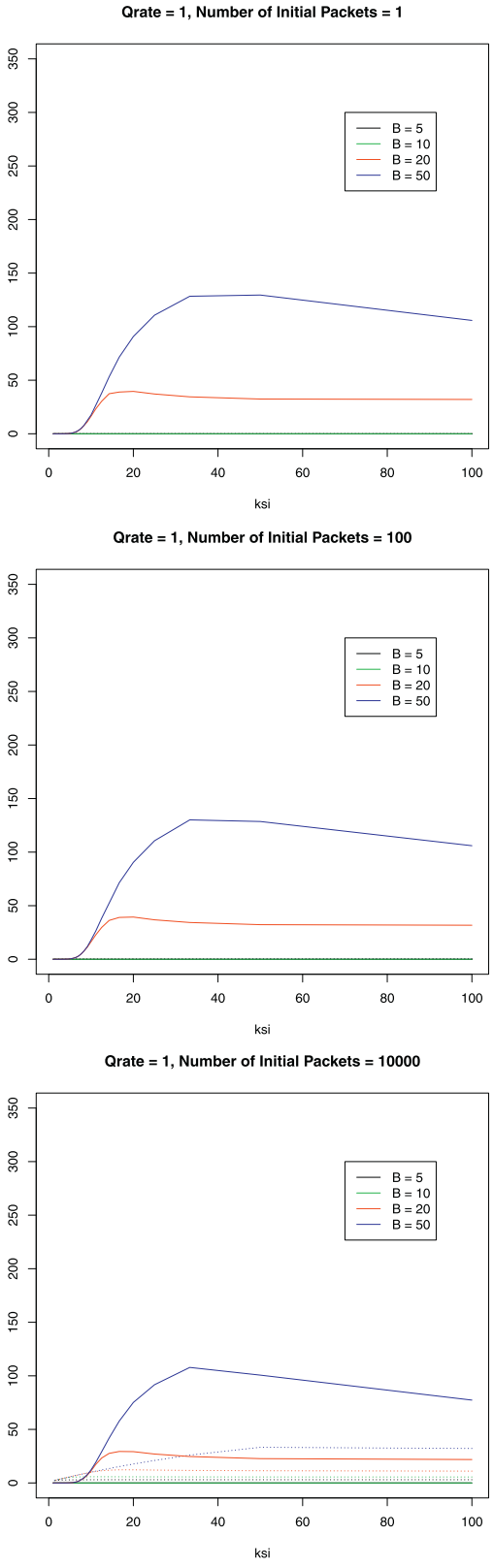

Fig. 8. The average lag time (in s) and initial delay (dotted lines) experienced by customers (with Qrate = 1).

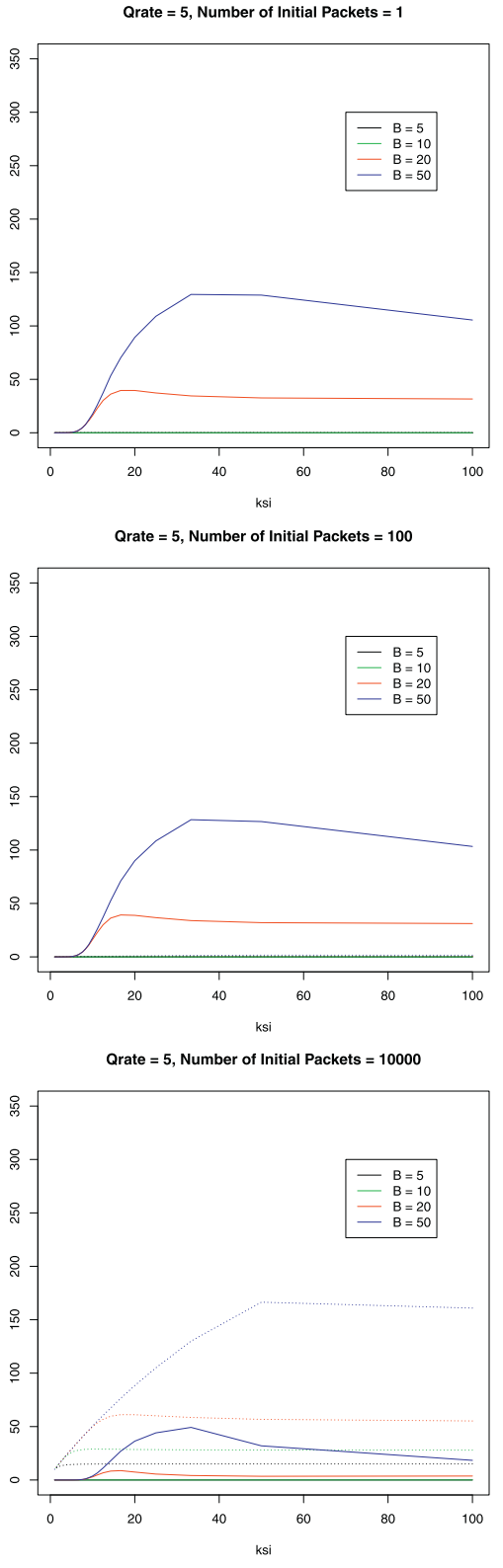

Fig. 9. The average lag time (in s) and initial delay (dotted lines) experienced by customers (with Qrate = 5).

interruption, as well as the peaks that indicates the worst case quality served by the system despite the requirements of arrivals.

First, B, the upper bound on number of customers that a virtual machine can handle simultaneously, can affect both the values of lag time and number of interruptions. As shown in all the plots in Figs. 8–10, in general, lag time (so as the number of interruptions) drops when B decreases from 50, 20, 10 to 5. It also shows that system developers can improve the worst service quality (the peak) by decreasing the value of B. On the other hand, there exists some payoff when system developers decrease the value of B. According to our

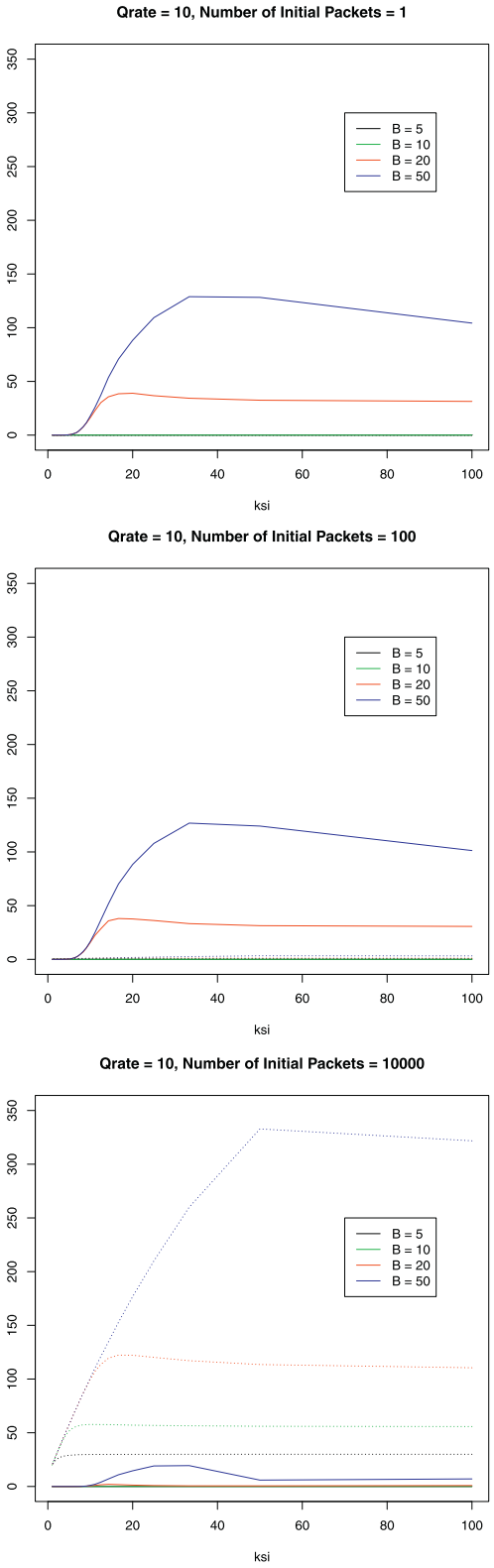

Fig. 10. The average lag time (in s) and initial delay (dotted lines) experienced by customers (with Qrate = 10).

queuing analysis, the increasing rate of number of workers (against the value of ξ) increases when B decreases (as shown in Fig. 6). It is needed system developers allocate sufficient resources to meet the requirement.

Second, N_{init}, the number of initial packets that the device on the client side collects before displaying, may also affect lag time and the number of interruptions. The first, second and the third diagrams in Figs. 8–10 show that the lag time after the accumulation of N_{init} sets of packets (solid lines) drops when N_{init} increases significantly from 1 or 100 to 10000. On the other hand, the initial delay (the dotted lines) increases since it takes longer for the device to collect more sets of packets. In many services, this waiting time on collecting sufficient initial packets can be fulfilled with advertisements from local services, and are used as means to prevent frequent interruptions and lag experienced. Third, Δ (or *QRate*), the quantum duration (or the factor for quantum duration) on the server side, may affect lag time when N_{init} is relatively large. The lag time has different behavior as N_{init} varies. For relatively small N_{init}, the lag time may remain the same when Δ (or *QRate*) increases (as shown on the first diagram in Figs. 8–10); as N_{init} increases, the initial delay may increase significantly for large *QRate* (as shown in the case $N_{init} = 10000$ and $QRate = 10$), and the lag time can decrease significantly (as shown in the third diagram in Figs. 8–10).

To sum up, we have the following properties of the presented proposed system according to the simulation results.

- The graphs of *L, LagTime* (so as *Interruption*) are similar in shape, and each shows a peak as ξ increases.
- The value of the peak is affected by the value of *B,* N_{init}, and Δ (or *QRate*).

Results in Figs. 6, 8, 9, and 10 provide the basis to select the most appropriate set of parameter values. Fig. 6 relates the number of workers required as a function of B. For a given amount of resource, i.e., value of B, Figs. 8–10 suggest values of *QRate* and N_{init} that provide acceptably small average lag time (we can also infer the average number of interruptions). It is possible (with sufficient resource on VMs) that under the assumptions of the system and environments, customers can experience high-standard service quality despite how busy the system is.

5. Conclusion

As online streaming services significantly increase in recent years, it becomes a critical issue to offer quality service via systems that benefit from cloud computing developments. We propose a study on simulating cloud based streaming services, combining queuing models and conditional simulation. This leads to a systematic approach to estimate resource provision of multi streaming services in the designed cloud system with respect to service quality of each streaming service under given arrival/service ratios and network environments. While most cloud simulation tools address discrete event systems, our hybrid approach provides a new insight to simulate streaming services in an efficient but yet precise way by separating server side and customer side behavior modeling at the packet-level.

The proposed approach provides managers of online streaming services a formal and systematic way to evaluate service quality without launching the service. In particular, we pioneer the study on quantitative analysis to estimate the personal experience of system customers for a model that incorporates the detailed physical layer of a cloud-based service platform. Under mild assumptions, our simulation approach provides exact simulation results for the whole system with multiple VMs by simulating the experience of individual customers on a single VM. The simulation results show how the service quality is affected by server and customer performance, providing the insight to select the system parameter values such as the maximum number of customers simultaneously served by a (working) VM and the time quantum assigned for each task in the VM, as well as the customer parameter values such as the initial delay before playing the video.

Our approach can be applied to suit different qualities of services. From historical data, the managers of an online streaming service can deduce the demand patterns of various streaming services for a particular time slot. Then the proposed approach can be applied to each type of streaming service to deduce the configuration of VMs; the total

amount of resource required is then deduced accordingly. In practice the presented service framework and analysis can be applied to other services where a cluster is employed to deal with bottleneck tasks. One of our ongoing work is applying the presented approach to measuring and adjusting the service quality of the streaming service [45] of National Palace Museum (NPM) in Taiwan.

Acknowledgment

This work was funded by the Ministry of Science and Technology, Taiwan, under the grants MOST 102-2420-H-004-006-MY2, 103-2221-E-001-028-MY3, 103-2221-E-004-006-MY3, 104-2410-H-004-123-, 105-2923-E-002-016-MY3, and 106-2221-E-004-002.

References

[1] V.K. Adhikari, S. Jain, Z. Zhang, Youtube traffic dynamics and its interplay with a tier-1ISP: an ISP perspective, Proceedings of the Tenth Annual Conference on Internet Measurement, (2010), pp. 431–443.
[2] Blinkx, Blinkx Buys Burst Media to Boost WebTV, Technology Hub Financeanc Times, 2011.
[3] B. Com, Global Digital Media? Entertainment a Key Online Activity, Budde Communications Annual Publication, 2012.
[4] P.d.Z. Bermudez, S. Kotz, Parameter estimation of the generalized Pareto distribution, J. Stat. Plan. Inference 140 (6) (2010) 1353–1373.
[5] S. Balsamo, V.D. Persone, P. Inverardi, A review on queueing network models with finite capacity queues for software architectures performance prediction, Perform. Eval. 51 (2–4) (2003) 268–288.
[6] V. Brazauskas, R. Serfling, Robust and efficient estimation of the tail index of a single-parameter Pareto distribution, North Am. Actuar. J. 4 (4) (2000) 12–27.
[7] N. Bouten, R.d.O. Famaey, J. Famaey, S. Latré, A. Pras, F. Turck, Qoe-driven in-network optimization for adaptive video streaming based on packet sampling measurements, Int. J. Comput. Telecommun. Netw. 81 (C) (2015) 96–115.
[8] K.T. Chen, Y.C. Chang, P.H. Tseng, C.Y. Huang, C.L. Lei, Measuring the latency of cloud gaming systems, Proceedings of the Nineteenth ACM International Conference on Multimedia, (2011), pp. 1269–1272.
[9] K.T. Chen, Y.C. Chang, H.J. Hsu, D.Y. Chen, C.Y. Huang, C.H. Hsu, On the quality of service of cloud gaming systems, Proceedings of the IEEE Transactions on Multimedia, 16 (2014). Feb, 2014
[10] H.K. Chu, W.P. Chen, F. Yu, Simulating time-varying demand services with queuing models, Proceedings of the Thirteenth IEEE International Conference on Services Computing, (2016), pp. 609–616.
[11] L. Cherkasova, FLEX: load balancing and management strategy for scalable web hosting service, Proceedings of the IEEE Symposium on Computers and Communications, (2000), pp. 8–13.
[12] H. Casanova, A. Legrand, M. Quinson, Simgrid: a generic framework for large-scale distributed experimentations, Proceedings of the Tenth IEEE International Conference on Computer Modelling and Simulation, (2008), pp. 126–131.
[13] R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A.F. De Rose, R. Buyya, Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Softw. Pr. Exp. (SPE) 41 (1) (2011) 23–50.
[14] M.E. Crovella, M.S. Taqqu, A. Bestavros, Heavy-tailed probability distributions in the world wide web, Proceedings of the A Practical Guide to Heavy Tails: Statistical Techniques and Applications, (1998), pp. 3–25.
[15] L. Chen, Y. Zhou, D.M. Chiu, A study of user behavior in online VoD services, Comput. Commun. 46 (2014) (2014) 66–75.
[16] L. Chen, Y. Zhou, D.M. Chiu, Smart streaming for online video services, IEEE Trans. Multimed. 17 (4) (2015) 485–497.
[17] S. Desmet, B. Volckaert, F.D. Turck, Design of a service oriented architecture for efficient resource allocation in media environments, Future Gener. Comput. Syst. 28 (3) (2012) 527–532.
[18] A.K.S.H.S. Dashti, R. Zimmermann, S.H. Kim, Streaming Media Server Design, Prentice Hall Professional Technical Reference, 2003.
[19] G-Cluster. http://www.g-cluster.com/en/index.html.
[20] L.V. Green, P.J. Kolesar, On the accuracy of the simple peak hour approximation for Markovian queues, Manag. Sci. 41 (8) (1995) 1353–1370.
[21] L.V. Green, P.J. Kolesar, W. Whitt, Coping with time-varying demand when setting staffing requirements for a service system, Proceedings of the Production and Operations Management, 16 (2007), pp. 13–39.
[22] V. Gupta, M.H. Balter, K. Sigman, W. Whitt, Analysis of join-the-shortest-queue routing for web server farms, Perform. Eval. 64 (2007) 1062–1081.
[23] T.P. Hettmansperger, M.A. Keenan, Tailweight, statistical inference and families of distributions - a brief survey, Stat. Distrib. Sci. Work 17 (1975) 161–172.
[24] R. Ibrahim, P. L'Ecuyer, Forecasting call center arrivals: fixed-effects, mixed-effects, and bivariate models, Manuf. Serv. Oper. Manag. 15 (1) (2013) 72–85.
[25] A. Ishii, T. Suzumura, Elastic stream computing with clouds, Proceedings of the IEEE International Conference on Cloud Computing, (2011), pp. 195–202.
[26] X. Kang, H. Zhang, G. Jiang, H. Chen, X. Meng, K. Yoshihira, Measurement, modeling, and analysis of internet video sharing site workload: a case study, Proceedings of the IEEE International Conference on Web Services, (2008).
[27] L. Kleinrock, Computer Applications - Queueing Systems, Volume 2 Edition, Wiley-Interscience, 1976.
[28] R. Krishnan, H.V. Madhyastha, S. Srinivasan, S. Jain, A. Krishnamurthy, T. Anderson, J. Gao, Moving beyond end-to-end path information to optimize CDN performance, Proceedings of the Ninth ACM SIGCOMM Conference on Internet Measurement, IMC, 2009, pp. 190–201.
[29] V. Krishnamoorthi, N. Carlsson, E. Halepovic, E. Petajan, BUFFEST: predicting buffer conditions and real-time requirements of HTTP(S) adaptive streaming clients, Proceedings of the Eighth ACM on Multimedia Systems Conference (MMSys'17), ACM, New York, NY, USA, 2017, pp. 76–87.
[30] P. L'ecuyer, J.H. Blanchet, B. Tuffin, P.W. Glynn, Asymptotic robustness of estimators in rare-event simulation, ACM Trans. Model. Comput. Simul. 20 (1) (2010) 1–41.
[31] Z. Li, M.A. Kaafar, K. Salamatian, G. Xie, Characterizing and modeling user behavior in a large-scale mobile live streaming system, IEEE Trans. Circuits Syst. Video Technol. 27 (2017) 2675–2686. ISSN 1051–8215.
[32] S. Latre, K. Roobroeck, T. Wauters, F.D. Turck, Protecting video service quality in multimedia access networks through PCN, IEEE Commun. 49 (12) (2011) 94–101.
[33] R. Nossenson, H. Attiya, The distribution of file transmission duration in the web, Proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems, (2003), pp. 647–654.
[34] OnLive. http://www.onlive.com.
[35] A. Pirkowski, A. Kempny, A. Hajduk, J. Strzelczyk, Load balancing for heterogeneous web servers, Proceedings of the Communications in Computer and Information Science, 79 (2010), pp. 189–198.
[36] Z. Qian, Y. He, C. Su, Z. Wu, H. Zhu, T. Zhang, L. Zhou, Y. Yu, Z. Zhang, Timestream: reliable stream computation in the cloud, Proceedings of the Eighth ACM European Conference on Computer Systems, (2013), pp. 1–14.
[37] P.J. Rasch, A queueing theory study of round-robin scheduling of time-shared computer systems, J. ACM 17 (1) (1970) 131–145.
[38] A. Riska, E. Smirni, G. Ciardo, Exact analysis of a class of GI/g/1-type performability models, IEEE Trans. Reliab. 53 (2) (2004) 238–249.
[39] M. Sakata, S. Noguchi, J. Oizumi, An analysis of the M/g/1 queue under round-robin scheduling, Operations Research, (1971), pp. 371–385.
[40] A. Silberschatz, G. Gagne, P.B. Galvin, Operating System Concept, seventh ed., Addison-Wesley, 2004.
[41] J.H. Son, M.H. Kim, An analysis of the optimal number of servers in distributed clientserver environments, Decis. Support Syst. 36 (3) (2004) 297–312.
[42] S.G. Steckley, S.G. Henderson, The error in steady-state approximations for the time-dependent waiting time distribution, Stoch. Models 23 (2) (2007) 307–332.
[43] StreamMyGame. http://www.streammygame.com/smg/index.php.
[44] R. Tewari, R. Mukherjee, D.M. Dias, H.M. Vin, Design and performance tradeoffs in clustered video servers, Proceedings of the International Conference on Multimedia Computing and Systems, (1996), pp. 144–150.
[45] R.H. Tsaih, Q.P. Lin, T.S. Han, Y.T. Chao, H.C. Chan, New ICT-enabled services of national palace museum and their designs, Proceedings of the Seventeenth International Conference on Cultural Economics, Kyoto, Japan, 2012.
[46] P. Velho, L.M. Schnorr, H. Casanova, A. Legrand, On the validity of flow-level tcp network models for grid and cloud simulations, Proceedings of the ACM Transaction Modeling and Computer Simulation, 23(4) (2013).
[47] S. Wee, H. Liu, Client-side load balancer using cloud, Proceeding of the SAC '10 Proceedings of the ACM Symposium on Applied Computing, (2010), pp. 399–405.
[48] M. Wieczorek, A. Hoheisel, R. Prodan, Towards a general model of the multi-criteria workflow scheduling on the grid, Future Gener. Comput. Syst. 25 (3) (2009) 237–256.
[49] R.W. Wolff, Poisson arrivals see time averages, Oper. Res. 30 (2) (1982) 223–231.
[50] Y. Xu, Z. Xiao, H. Feng, T. Yang, B. Hu, Y. Zhou, Modeling buffer starvations of video streaming in cellular networks with large-scale measurement of user behavior, Proceedings of the IEEE Transactions on Mobile Computing, 16 (2017), pp. 2228–2245.
[51] H. Yin, X. Liu, T. Zhan, V. Sekar, F. Qu, C. Lin, H. Zhang, B. Li, Design and deployment of a hybrid CDN-P2P system for live video streaming: experiences with Livesky, Proceedings of the Seventeenth ACM International Conference on Multimedia, (2009), pp. 25–34.
[52] F. Yu, Y.W. Wan, R.H. Tsaih, Quantitative analysis of cloud-based streaming services, Proceedings of the IEEE International Conference on Services Computing, (2013), pp. 216–223.