Cloud provisioning in the QED regime

Johan S.H. van Leeuwaarden
Eindhoven University of Technology
j.s.h.v.leeuwaarden@tue.nl

Britt W.J. Mathijsen
Eindhoven University of Technology
b.w.j.mathijsen@tue.nl

Fiona Sloothaak
Eindhoven University of Technology
f.sloothaak@tue.nl

ABSTRACT

We design a dynamic algorithm for dimensioning and stabilizing a cloud provisioning process. We model the process as a semi-open tandem network consisting of a multi-server queue with host servers that respond to new requests of cloud users and an infinite-server queue that contains the active cloud users. The algorithm matches load predictions by jointly setting the number of host servers and the total available capacity. This dual setting is time-dependent and based on the modified offered load (MOL) method. The algorithm is made to perform in the Quality-and-Efficiency-Driven (QED) regime to achieve economies of scale, and is equipped with the feature of repeated requests, which lets initially blocked users retry to get access to the cloud system after a certain delay. Extensive numerical simulations show that the algorithm stabilizes performance at high QoS-levels, even in face of strongly time-varying loads and repeated requests.

CCS Concepts

•**Networks** → *Cloud computing;* •**Mathematics of computing** → **Queueing theory;**

Keywords

stochastic models, queues and queueing networks, cloud computing systems, heavy traffic, asymptotic analysis

1. INTRODUCTION

Cloud computing enables network access to a shared pool of configurable computing resources, allowing users (e.g. companies, service providers) to store and process their data in third-party data centers, without investing in the operating equipment themselves. At the foundation of cloud computing lies the idea of sharing resources to achieve economies of scale in terms of maximizing computing power usage and reducing the overall cost of resources such as energy and infrastructure. Cloud providers, such as Amazon EC2, Windows Azure and Rackspace [2], offer virtual machine (VM) provisioning, which allows users to request VM instances configured to their preference. In a service system context, the provider thus serves users by supplying them with a VM that matches their requirements, running on one of the cloud's physical machines.

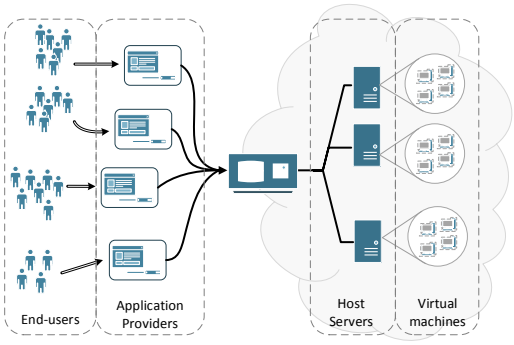


Figure 1: Cloud provisioning process

In a well-managed cloud, the available computing resources are seemingly unlimited and the user is able to almost instantaneously increase or decrease its computational capacity. This ability to dynamically adapt capacity to cover for unanticipated fluctuations in workload, commonly referred to as *elasticity*, sets apart cloud computing from conventional computing paradigms such as grid computing [2, 15]. From the cloud provider's viewpoint, however, the capacity is naturally finite: only a certain number of VM instances can be hosted simultaneously. It is essential for the provider's credibility to design its cloud such that the users experiences high quality-of-service (QoS) levels [5, 16]. On the other hand, over-provisioning causes expensive running costs of idle servers, and is therefore undesirable. We will provide a tractable model that captures the dominant dynamics of the VM provisioning process. Based on asymptotic analysis of this model, we gain insights that lead the way towards efficient capacity allocation in cloud systems that balances both objectives, even in the realistic but challenging scenario of time-varying demands. Let us describe the cloud provisioning process in more detail; see Figure 1. At the highest granularity level there are the *end-users*, devices typically directly operated by humans, using an *application provider* (AP), say a company that provides software usage over the internet (e.g. SaaS [15]). To some extent, the AP will rely on a static set of computing resources, but

VALUETOOLS 2015, December 14-16, Berlin, Germany

DOI 10.4108/eai.14-12-2015.2262649

certainly in case of sudden surges in workload, these might not be sufficient. When the AP recognizes the need for additional capacity, e.g. by *auto-scaling* procedures [1], a VM request is submitted to the cloud provider. The request is handled by a *host server* that starts the set-up of the VM with requested specifications. This includes elementary operations such as copying the VM image and assigning an IP address. Each server is able to host multiple VM instances in parallel, although the VMs in set-up need their dedicated attention, due to concurrency level constraints incurred by large I/O activities. Once the set-up is completed, the VM is ready for use, and the AP may start using the additional computing resources.

Our focus lies on the capacity allocation within the cloud environment, so the right side of Figure 1. Successful management of cloud systems requires the right scaling of both the number of host servers (denoted by s) at the first I/O queue and the maximum number of VMs (denoted by n) that can be hosted simultaneously. Moreover, this needs to be done in a dynamic way in order to respond effectively to the time-varying demand. The capacity n defines a hard constraint on whether a new VM request will be accepted immediately or not. Therefore, new requests will be delayed or even dropped if the available host capacity is insufficient, which is more likely to occur during periods in which the s host servers are overloaded.

To describe the cloud system in mathematical terms, we use the model developed in [17]. Each host server may host a number of VM instances at the same time, yielding a total number of n parallel VM instances. Requests, arriving to the system according to a Poisson process with rate λ, are granted only if one of these n positions is available. If granted, the request is assigned to a host server not busy initializing another VM instance, if available, or waits for one to become available. This start-up time is assumed to be exponentially distributed with mean $1/\mu$. On completion of the initialization phase, VM usage is initiated by the client. The VM continues to be occupied for a random amount of time, with mean $1/\theta$, until release by the user. We note that the model in [17] has three queues in tandem, one $M/M/s$ queue, followed by two $M/M/\infty$ queues that separately model a second initialization phase and the actual VM usage by the cloud user. We thus replace the two $M/M/\infty$ queues by one $M/G/\infty$ queue with an aggregated service time, which does not alter the overall system performance. Under the above Markovian assumptions, we obtain the semi-open Jackson network in Figure 2. The model

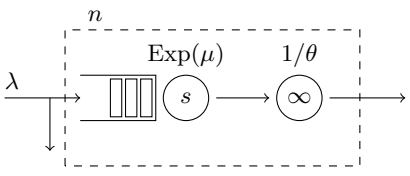


Figure 2: Abstracted model of VM provisioning process.

is able to capture accurately how the two factors s and n limit the scalability, even when the results of the model are applied to (non-Markovian) real data traces; see [17] for extensive data experiments. Due to the insensitivity of the $M/G/\infty$ queue with respect to the service time distribution, the cloud model can capture all distributions for actual cloud usage times that fit empirical measurements. Hence, for the results presented in this paper in terms of $1/\theta$, one can interpret $1/\theta$ as the mean of some generally distributed (realistic) usage time.

Our contribution consists in developing an effective algorithm to determine the two critical resource levels s, n to stabilize performance at the first multi-server queue. We do so by setting a QoS-target, with the delay probability at the multi-server queue as the leading example, and then design a dynamic algorithm for setting s and n that lead to stable system behavior, around some target, at all times. The algorithm contains three essential elements: (i) It is made to perform in the Quality-and-Efficiency-Driven (QED) regime to achieve economies of scale; (ii) It uses the modified offered load (MOL) method to account for time-varying loads; (iii) It is equipped with the feature of repeated requests, which essentially lets initially blocked users retry to get access to the cloud system after a certain delay. Next, we will discuss each of these three elements in more detail.

1.1 QED regime

Our goal is to design an algorithm that ensures high resource utilization, while at the same time maintains high QoS, which is perfectly aligned with the philosophy behind the QED regime known in many-server asymptotic theory. Let $R = \lambda/\mu$ be the workload offered to the system and let $\rho = R/s$ denote the average utilization level of the host servers. In the QED regime [4, 8], the utilization level is driven to unity in accordance to $(1-\rho)\sqrt{s} \to \beta$ as $s \to \infty$, for some fixed parameter $\beta > 0$. This gives rise to the square-root scaling rule $s = R + \beta\sqrt{R}$, which prescribes in the cloud setting (ignoring the role of n for now) that the number of host servers s should exceed the minimally required offered load R, but only by a relatively small number $\beta\sqrt{R}$. As s grows large, the probability of delay tends to a non-degenerate function that only depends on the parameter β. This function is strictly decreasing in $\beta > 0$ with range $(0, 1)$. Consequently, any targeted delay probability can be achieved by the adjustment of β. Moreover, the mean delay is of order $1/\sqrt{s}$ and hence asymptotically negligible. Since cloud environments typically consist of thousands of servers, this asymptotic framework is particularly relevant.

The semi-open tandem network in Figure 2 can be described as a closed Jackson network. The stationary distribution therefore obeys a product-form, which was exploited in [17, 11] to obtain the QED limit behavior. Interestingly, the QED scaling rules can be expressed in a two-fold QED scaling policy: $s = R_s + \beta\sqrt{R_s}$ for the number of host servers and $n = R_n + \eta\sqrt{R_n}$ for the number of VM positions, for some constants $\beta, \eta > 0$, and with R_s and R_n the offered load to s and the offered load to n, respectively. This is similar to the two-fold QED scaling rules that were applied to other models in [9, 21]. Under the two-fold scaling policy, limiting QED expressions are derived for stationary performance, such as the delay probability and the mean delay, in the form of explicit functions evaluated in both β and η.

1.2 Dynamic algorithm

As the cloud provider's equipment consists of a fixed number of servers, the maximum load the cloud can handle is upper bounded. However, constantly keeping all of these servers up and running, e.g. matching the daily peak workload, leads to severe over-provisioning during more quiet periods. Tremendous power savings can be achieved by turning

some of the physical machines to sleep mode during these quiet periods. Therefore, there is a growing need to develop algorithms that exploit the potential improvement in the energy efficiency and utilization of resources of the massive data centers behind cloud computing [13, 20]. This needs to be done such that the available resources not only match the time-varying demand, but also meet QoS-targets.

There are two commonly used approaches to address this problem of time-dependent loads in the QED context. The first relies on using steady-state approximations, such as the Piecewise Stationary Approximation (PSA). This method divides the time-horizon into small intervals, predicts the expected work load for each interval and applies the QED square-root rule. However, this approximation is accurate only when usage times are short relative to the rate of change, an assumption that is typically not satisfied in cloud environments [17]. We will therefore resort to the second common approach: the modified offered load (MOL) method. We will transform the two-fold QED rules into a time-varying counterpart based on the MOL method that approximates the offered load at both stages of the cloud system via a corresponding system with ample resources. For capacity planning to meet uncertain exogenous demand, the MOL method thus looks at the amount of capacity that would be used if there were no constraints on its availability (taking s and n both infinite). This simplification results in a tandem network consisting of two infinite-server nodes for which the offered load becomes tractable. Since the mean delay is asymptotically negligible in the QED regime, the offered load in the infinite-capacity system provides excellent approximations, which can serve as input for our algorithm that sets capacity levels to stabilize performance, even in face of a possibly strongly time-varying arrival rate and the tandem network structure. Apart from stabilizing performance, we also show that our algorithm can maintain *high* QoS-levels, despite the potentially enormous traffic volumes, so that it fully exploits both the elasticity (MOL method) and potential economies of scales (QED regime) of cloud systems.

1.3 Repeated requests

The dominant assumption in the literature is to discard the requests that meet a full system upon arrival. From a practical viewpoint, though, it seems more plausible that blocked cloud users prefer to repeat their request. To accommodate this, we introduce the feature of *repeated requests* to the model in Figure 2: Users that meet a full system (n users) upon arrival are no longer discarded, but instead are allowed to reattempt getting access after a stochastic delay, until they are successful. This distributed policy brings several advantages. First, all users are (ultimately) served. Second, the 'virtual' queue of initially blocked users requires no additional overhead for the cloud provider. Third, there is potential workload shifting. Requests are typically blocked during periods of overload. By shifting the requests ahead in time, the user likely meets a less congested system, so that the overall workload in the cloud system is distributed more evenly over time. This idea was recently developed by [19] in the context of the $M/M/s/n$ queue as a way to temporarily release pressure from the system during overloaded periods.

The form of reattempts considered in this paper lets blocked users reattempt after exponential times, which is just one example of a broad range of different mechanisms that can be designed. One could for instance distinguish between fully distributed and more advanced approaches, depending on whether the initiative resides with the cloud system actively approaching blocked users, or with the blocked users that independently try to enter the system. It is clear that the exponential reattempts make the system distributed and easily implementable. It also requires little communication overhead, which is particularly relevant in large-scale data centers that deploy thousands of servers and handle massive demands with service requests coming in at huge rates.

The remainder of this paper is structured as follows. In Section 2 we present the main ideas related to QED scalings, repeated requests and the MOL method. In Section 3, we apply these ideas to the cloud system, and construct a dynamic dimensioning algorithm. Section 4 presents extensive numerical results to assess the performance of our algorithm and we conclude in Section 5.

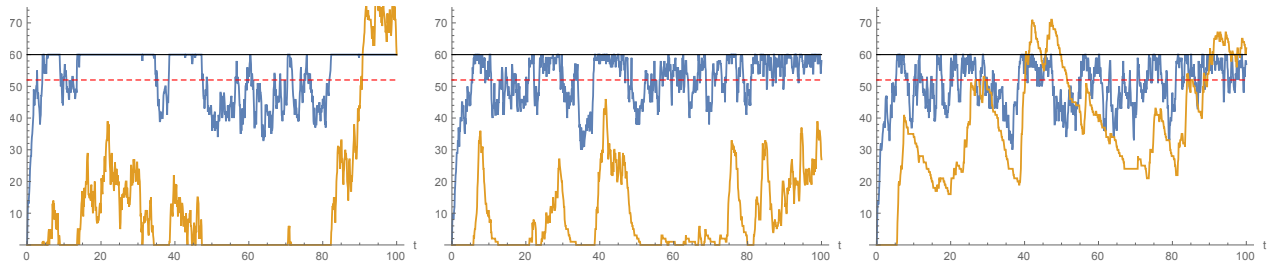
2. MAIN IDEAS

In this section we introduce and demonstrate the main ideas in this paper by applying them to an $M/M/s/n$ system, presented in the recent work [19]. This model is a simplified version of the cloud system in Figure 2 that arises when $\theta \to \infty$. Note that large θ-values are not realistic in cloud systems, for which the most likely scenario is $\theta \ll \mu$. Therefore, this section serves the purpose of conveying the general concepts, while in Section 3 we apply these concepts to the cloud system, also in the regime $\theta \ll \mu$.

The $M/M/s/n$ system with reattempts can be described in terms of s servers to which users arrive according to a Poisson process with rate λ. A user that finds a free server upon arrival occupies this server immediately, while users that meet more than s but fewer than n users in the system are admitted but wait for an available server. Users that meet n occupied VMs upon arrival are not admitted directly, but reattempt to get access after an exponential time with mean $1/\delta$. Each initially blocked user performs reattempts until admitted. Service times are exponentially distributed with mean $1/\mu$, and interarrival times, service times and reattempt times are mutually independent. The system state can then be described as a two-dimensional process $\{(C(t), N(t)); t \geq 0\}$ with $C(t)$ the number of users assigned to a VM and $N(t)$ the number of users that perform reattempts. Under the above assumptions this process is a continuous-time Markov chain on $\{0, 1, \ldots, n\} \times \{0, 1, \ldots\}$.

To provide intuition for the effect of reattempts, Figure 3 depicts typical sample paths of the process describing the number of users present in the system $X(t) := C(t) + N(t)$, for three different values of δ. To make a fair comparison, we coupled the arrival processes of the three simulations. As expected, the sample paths confirm that the impact of reattempts increases with the mean delay time $1/\delta$. The process $X(t)$ spends a considerable fraction of time below level s, which is indicative of a small delay probability or, more generally, high QoS. Note also that for $\delta = 0.1$ the process $X(t)$ is concentrated, and only mildly fluctuates, around the optimal point of operation s. Small delay probabilities and operating near full occupancy level are distinguishing features of the QED regime, and we see that these features are even more distinctive with reattempts.

We assume that $R = \lambda/\mu < s$, which is a necessary and sufficient condition for ergodicity (see [7, Chapter 2]). Reat-



(a) $\delta = 1000$ (b) $\delta = 1$ (c) $\delta = 0.1$

Figure 3: Sample paths of $X(t)$ (blue) and $N(t)$ (orange) for $R = 50$, $\beta = 0.2, \eta = 1$. s and n are indicated by the red and black lines, respectively.

tempts thus create a process of retrials, which is typically quite challenging to analyze [7]. Even for deriving the stationary distribution of retrial systems one often resorts to computational approaches [3]. These numerical approaches face increasing numerical difficulties when the number of servers grows large. The QED regime for this model concerns the two-fold scaling

$$s = R + \beta\sqrt{R}, \quad \beta > 0, \tag{1}$$

$$n = s + \eta\sqrt{s}, \quad \eta > 0 \tag{2}$$

and we focus on QED approximations for the stationary delay probability $P_r(\text{delay})$ as s, n, R become large.

When the blocked users are simply discarded, the process $\{(C(t), N(t)); t \geq 0\}$ reduces to a standard $M/M/s/n$ system. This is a birth-death process with an elementary characterization of the stationary distribution $\{\pi_i\}$ with π_i the stationary probability of i users in the system. The blocking probability is the probability that an arriving user is discarded from the system, and the delay probability corresponds to the probability that an admitted user finds all s hosts occupied upon arrival. Due to the PASTA property, the delay probability and blocking probability are then given by $P(\text{delay}) = \sum_{i=s}^{n-1} \pi_i/(1-\pi_n)$ and $P(\text{block}) = \pi_n$. Under the QED scaling (1)–(2), when n, s and R grow to infinity simultaneously while β and η are fixed, we have [14]

$$P(\text{delay}) \to \frac{1 - \mathrm{e}^{-\beta\eta}}{1 - \mathrm{e}^{-\beta\eta} + \beta\Phi(\beta)/\phi(\beta)} =: g(\beta, \eta), \tag{3}$$

$$\sqrt{R} \cdot P(\text{block}) \to \frac{\beta \mathrm{e}^{-\beta\eta}}{1 - \mathrm{e}^{-\beta\eta} + \beta\Phi(\beta)/\phi(\beta)} =: f(\beta, \eta). \tag{4}$$

When the time between reattempts is relatively long compared with the service time, the process of reattempting users becomes roughly Poisson. Due to (4), the mean additional load due to reattempts must be of order $\sqrt{R}$. We can thus assume that the total arrival rate R_{tot} takes the form $R_{\text{tot}} = R + \alpha\sqrt{R}$ for some $\alpha > 0$. Then (1) is asymptotically equivalent with

$$s = R_{\text{tot}} + (\beta - \alpha)\sqrt{R_{\text{tot}}}, \tag{5}$$

while the scaling rule in (2) remains unchanged. Therefore, the retrial system with QoS parameter β mimics an $M/M/s/n$ system with parameter $\beta_\alpha = \beta - \alpha$. Note that the volume of blocked users in this setting is $f(\beta - \alpha, \eta)\sqrt{R_{\text{tot}}}$. This quantity must equal the mean additional load $\alpha\sqrt{R} \sim \alpha\sqrt{R_{\text{tot}}}$ and therefore we obtain the fixed point equation

$$\alpha = f(\beta - \alpha, \eta). \tag{6}$$

Numerically determining α is straightforward, particularly because it is uniquely defined (see [19, Lemma 1]). As a result, the delay probability $P_r(\text{delay})$ in the model with reattempts $\{(C(t), N(t)); t \geq 0\}$ can be approximated in the QED regime (1)–(2) by

$$P_r(\text{delay}) \approx g(\beta - \alpha, \eta). \tag{7}$$

The asymptotic QED expressions for the system *without* reattempts given in (3) and (4) together with the corrections obtained through (6) thus provide a method for dimensioning an $M/M/s/n$ system *with* reattempts. For sufficiently large arrival volumes, we can tune the QoS-levels offered by the system through the QoS-parameters β and η. To demonstrate this, we choose the delay probability as a vehicle, and search for a pair (β, η) that realize a target delay probability ε. In a system without reattempts, attaining the target performance boils down to finding a pair $(\beta_\varepsilon, \eta_\varepsilon)$ such that $g(\beta_\varepsilon, \eta_\varepsilon) = \varepsilon$. It is easily verified that g is a decreasing function in β and an increasing function in η ranging from zero to one. Hence, for all targets ε there exists a solution to $g(\beta_\varepsilon, \eta_\varepsilon) = \varepsilon$, although this solution might not be unique. Nevertheless, by fixing the performance parameter η one can solve for the unique β_ε to achieve the targeted ε and dimension the number of hosts s accordingly. This can be extended to the more involved system with reattempts, by determining β^*_ε such that

$$g(\beta^*_\varepsilon, \eta) = \varepsilon, \tag{8}$$

before replacing β^*_ε by $\beta_\varepsilon - \alpha_\varepsilon$ with $\alpha_\varepsilon = f(\beta_\varepsilon - \alpha_\varepsilon, \eta)$, ultimately resulting in

$$\beta_\varepsilon = \beta^*_\varepsilon + f(\beta^*_\varepsilon, \eta). \tag{9}$$

Once a feasible pair $(\beta_\varepsilon, \eta)$ is obtained, (1) and (2) specify the capacity levels s and n that (asymptotically) achieve the target delay probability. Extensive simulations illustrate that this dimensioning scheme works well [19], particularly for small δ.

We next discuss how the parameters s and n can be adjusted in time-varying environments where the offered load $R(t)$ is a function of time. For this we use the MOL method, which was developed in [10] to approximate and dimension the $M_t/G/s$ system by establishing a relation with the analytically tractable $M_t/G/\infty$ system. An underlying assumption of the MOL method is that a well-capacitated multi-server queue delays only a small portion of users and only for short periods. Therefore, the system can be approximated by an infinite-server system. The MOL approximation [10] combines the desirable QoS properties rendered by the QED regime with the analytic tractability of the $M/G/\infty$ queue, see [6], to establish a dynamic algorithm for choosing $s(t)$

that stabilizes the system behavior at some QoS-target. This is the approach we shall take as well, first for the $M/M/s/n$ system below, and then for the cloud system in Section 3.

To understand why the MOL approximation is likely to be accurate for the systems in this paper, observe that under the scaling rules (1)–(2), the blocking probability vanishes asymptotically and hence the main assertion on which the MOL approximation is built continues to hold. Following the line of thought in [10], we consider the number of users in a system with $s = n = \infty$ to obtain $\hat{R}(t) = \mathbb{E}[R(t-S)]\mathbb{E}S$, where S is the service requirement per user taken to be unit exponentially distributed. Then,

$$\hat{R}(t) = \int_0^\infty e^{-u} R(t-u)\,\mathrm{d}u. \tag{10}$$

Note that this transformation typically shifts and levels peaks in workload ahead in time with respect to those in $R(t)$. As the time-varying counterparts of (1)–(2), we then get [19]

$$s(t) = \hat{R}(t) + \beta\sqrt{\hat{R}(t)}, \tag{11}$$

$$n(t) = s(t) + \eta\sqrt{s(t)}. \tag{12}$$

These rules apply to the situations with and without reattempts, although in the former case the QoS parameters β and η need corrections. In order to set $s(t)$ and $n(t)$ to achieve a target performance level, one needs to determine the required parameters β and η through selection of η and computation of β_ε as in (9). This yields a time-varying dimensioning scheme, which proves to be robust against heavy fluctuations in offered load. We will demonstrate the power of such dynamic schemes in Section 4, but then for a more advanced algorithm customized for the cloud system.

3. CLOUD SYSTEM

We now return to the cloud model in Figure 1 and apply the ideas introduced in Section 2 to this more advanced setting. As an exact analysis for this model becomes computationally infeasible quickly, as s and n grow large, we again resort to the asymptotic QED regime. Following the approach in [17, 11], we scale the systems parameters according to

$$s = R + \beta\sqrt{R}, \quad -\infty < \beta < \infty, \tag{13}$$

$$n = s + \frac{R}{\theta} + \eta\sqrt{\frac{R}{\theta}}, \quad \eta > 0. \tag{14}$$

Note that (14) implicitly assumes that R/θ, the mean number of users in the infinite-server queue, is the value around which the total number of VMs in the system will settle. This will certainly be the case when $\theta \ll \mu$, which indeed is the dominant scenario in cloud settings [17]. Under (13)–(14),

$$P(\text{delay}) \to \frac{\xi_1 - \xi_2}{\gamma + \xi_1 - \xi_2} =: g_C(\beta,\eta), \tag{15}$$

$$\sqrt{R}\cdot P(\text{block}) \to \frac{\nu}{\gamma + \xi_1 - \xi_2} =: f_C(\beta,\eta), \tag{16}$$

for $R \to \infty$, where $\kappa = \sqrt{\mu/\theta}$ and

$$\gamma = \int_{-\infty}^{\beta} \Phi(\eta + (\beta - t)\kappa)\varphi(t)\mathrm{d}t, \quad \xi_1 = \frac{\varphi(\beta)\Phi(\eta)}{\beta}, \tag{17}$$

$$\xi_2 = \frac{1}{\beta}\,\varphi(\sqrt{\beta^2+\eta^2})\exp\{\tfrac{1}{2}(\eta-\beta\kappa)^2\}\Phi(\eta-\beta\kappa), \tag{18}$$

$$\nu = \frac{1}{\sqrt{1+\kappa^2}}\varphi\left(\frac{\eta\kappa+\beta}{\sqrt{1+\kappa^2}}\right)\Phi\left(\frac{\beta\kappa-\eta}{\sqrt{1+\kappa^2}}\right) + \beta\,\xi_2. \tag{19}$$

Note that these expressions only involve the ratio of the service rates of the two queues. Therefore, without loss of generality, we henceforth set $\mu = 1$.

To let the feature of repeated requests comply with QED system behavior and to combine both features into one algorithm, we need to quantify the impact of reattempts in the QED regime. Again assume that blocked users reattempt after i.i.d. exponential times with mean $1/\delta$. A direct analytic approach is obstructed by the absence of a product-form solution in case of repeated requests. To see this, note that the dynamics of the system in Figure 2 can be described as a three-dimensional stochastic process $\{N(t), X(t), Q(t)\}$ with $N(t)$ the number of requests waiting to reattempt, $X(t)$ the number of admitted requests and $Q(t)$ the number of admitted requests at the multi-server queue. The process lives on the state space $\{0,1,...\}\times\{0,1,...,n\}\times\{0,1,...,n\}$ and is stable when $R < s$. Under this assumption, a stationary distribution exists. Determining the stationary distribution, however, should be done numerically, since the model does not belong to a class of networks with a closed-form solution.

Using the ideas from Section 2, we will argue that the cloud system with repeated requests resembles the standard system without reattempts, as in [11, 17], but with an increased load. By (16) we argue that for sufficiently large systems, the load that comes with reattempts is of the form $\Omega = \alpha\sqrt{R}$ for some $\alpha > 0$. Hence, the total load offered to the system *with* reattempts is roughly $R_{\text{tot}} = R + \alpha\sqrt{R}$, implying $R = R_{\text{tot}} - \alpha\sqrt{R_{\text{tot}}} + O(\sqrt{R_{\text{tot}}})$. As before, we rewrite (13)–(14) in terms of R_{tot} to get

$$s = R_{\text{tot}} + (\beta - \alpha)\sqrt{R_{\text{tot}}} + O(\sqrt{R_{\text{tot}}}) \tag{20}$$

and

$$\begin{aligned} n - s &= (R_{\text{tot}} - \alpha\sqrt{R_{\text{tot}}})\frac{1}{\theta} + \eta\sqrt{\frac{R_{\text{tot}}}{\theta}} + O(\sqrt{R_{\text{tot}}}) \\ &= \frac{R_{\text{tot}}}{\theta} + \left(\eta - \frac{\alpha}{\sqrt{\theta}}\right)\sqrt{\frac{R_{\text{tot}}}{\theta}} + O(\sqrt{R_{\text{tot}}}). \end{aligned} \tag{21}$$

Accordingly, the constant α is defined as the solution of the fixed point equation

$$\alpha = f_C(\beta - \alpha, \eta - \alpha/\sqrt{\theta}). \tag{22}$$

If we provision a large-scale cloud system according to the QED scaling rules (13)–(14), the cloud system *with* reat-

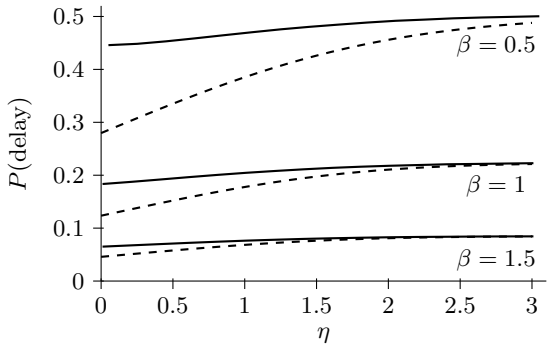


Figure 4: Asymptotic delay probability in the cloud model as function of η with (solid) and without (dashed) reattempts for several values of β.

	$(\beta,\eta)=(1,1)$				$(\beta,\eta)=(1,0.5)$				$(\beta,\eta)=(0.5,1)$			
			(b)	(r)			(b)	(r)			(b)	(r)
R	s	n	$\alpha=0$	$\alpha=0.133$	s	n	$\alpha=0$	$\alpha=0.327$	s	n	$\alpha=0$	$\alpha=0.303$
10	13	70	0.225	0.257	13	67	0.198	0.244	12	69	0.349	0.409
50	57	323	0.195	0.223	57	315	0.168	0.211	54	320	0.366	0.434
100	110	632	0.187	0.216	110	621	0.161	0.203	105	627	0.393	0.475
1000	1032	6103	0.177	0.203	1032	6067	0.152	0.191	1016	6087	0.384	0.467
5000	5071	30229	0.178	0.204	5071	30150	0.152	0.192	5035	30193	0.389	0.469
$g_C(\beta-\alpha,\eta-\alpha/\sqrt{\theta})$			0.1778	0.2043			0.1521	0.1939			0.3852	0.4687

Table 1: Comparison of approximated delay probabilities in the cloud model with $\theta = 0.2$ and $\delta = 10^{-2}$ against simulated values for finite-size systems. The values obtained through the asymptotic scheme are given in the bottom row.

tempts performs as a cloud system *without* reattempts but with the parameters (β,η) replaced by $(\beta-\alpha,\eta-\alpha/\sqrt{\theta})$. For instance,

$$P_r(\text{delay}) \approx g_C(\beta-\alpha,\eta-\alpha/\sqrt{\theta}) \tag{23}$$

for R large. The impact of reattempts on the delay probability is illustrated in Figure 4. Observe that the effect of reattempts decreases with β and diminishes with η.

The approximation (23) for an otherwise seemingly untractable system provides a crucial ingredient for the dynamic algorithm that we construct next. In the system without reattempts, one finds parameters β_ε and η_ε satisfying $g_C(\beta_\varepsilon,\eta_\varepsilon)=\varepsilon$ and computes the associated capacity levels by (13)–(14) accordingly. Dimensioning the cloud system with reattempts is performed in a similar manner as prescribed by (9). However, due to (14), the fixed point α appears in both arguments of the function f_C in (22). To obtain a unique solution for β_ε, we adjust the input variable η^* to complete the dimensioning framework. The details are given in Algorithm 1. We are then in a position to design

Input: Target delay probability $\varepsilon \in (0,1)$,
$\eta^* := \eta - \alpha/\sqrt{\theta} > 0$.
Output: Variability hedge β_ε and η_ε.

1. Compute β^* such that $g(\beta^*_\varepsilon,\eta^*)=\varepsilon$.
2. Set $\beta_\varepsilon = \beta^*_\varepsilon + f(\beta^*_\varepsilon,\eta^*)$ and $\eta_\varepsilon = \eta^* + f(\beta^*_\varepsilon,\eta^*)/\sqrt{\theta}$.

Algorithm 1: Stationary dimensioning for cloud model with reattempts.

an algorithm based on the QED regime with reattempts to achieve robust performance in time-varying environments. As before, let $R(t)$ represent the offered load at time t. Then the number of requests at the host server queue is approximately given by

$$R_1(t) = \int_0^\infty e^{-u} R(t-u)\,\mathrm{d}u. \tag{24}$$

Contrary to the $M/M/s/n$ system, the number of users present in the system strongly depends on the number of requests in the second phase of the system, especially since $\theta \ll \mu$. Therefore, we need an approximation for the workload offered to the second queue as a function of t, which we denote by $R_2(t)$. Continuing the reasoning of MOL, we argue that $R_2(t)$ is equal to the output process of the first queue $R_1(t)$. Then,

$$\begin{aligned} R_2(t) &= \mathbb{E}[R_1(t-S_2)]\mathbb{E}S_2 \\ &= \int_0^\infty \int_0^\infty e^{-u-\theta v} R(t-u-v)\mathrm{d}u\,\mathrm{d}v. \end{aligned} \tag{25}$$

Combining the above ingredients then leads to Algorithm 2, in which $[\cdot]$ denotes the integer rounding operator. Observe

Input: Target delay probability $\varepsilon \in (0,1)$,
$\eta^* := \eta - \alpha/\sqrt{\theta} > 0$.
Output: Time-dependent capacity levels $s(t)$, $n(t)$ achieving $P_r(\text{delay}) = \varepsilon$.

1. Compute β_ε and η_ε according to Algorithm 1 with input η^*.
2. Compute $R_1(t)$ and $R_2(t)$ as in (24) and (25).
3. Return

$$s(t) = [R_1(t) + \beta_\varepsilon \sqrt{R_1(t)}], \tag{26}$$

$$n(t) = [s(t) + R_2(t) + \eta_\varepsilon \sqrt{R_2(t)}]. \tag{27}$$

Algorithm 2: Dynamic dimensioning for cloud model with reattempts.

that if the service times at the infinite-server queue are relatively short compared with the rate of change of the load function, we have $R_2(t) \approx R_1(t)/\theta$, so that (26)–(27) shows resemblance with (13)–(14).

4. NUMERICAL RESULTS

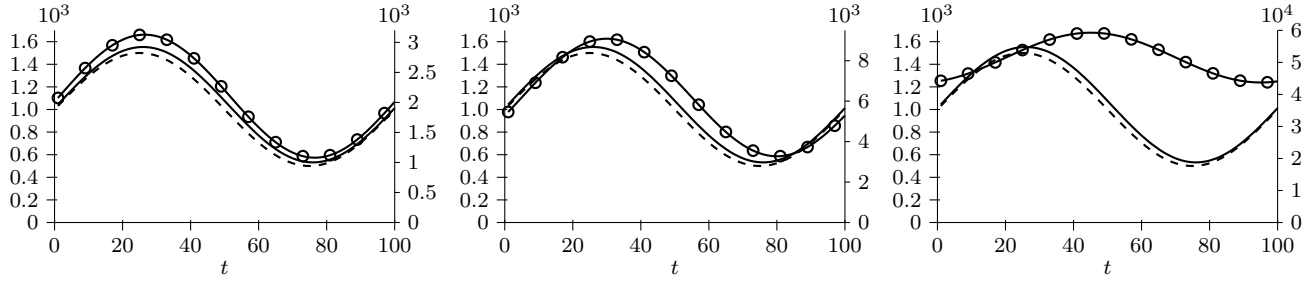
In this section we present extensive numerical experiments to demonstrate the excellent performance of the dynamic algorithm developed in Section 3 for the cloud system.

4.1 Accurate approximations

We assess the accuracy of the QED approximations that are obtained for $R \to \infty$ for finite systems. Column (b) in Table 1 compares $f_C(\beta,\eta)$ with the 'true' delay probabilities that follow from extensive simulations. All reported simulation results are accurate up to a 95% confidence interval of width $< 3 \cdot 10^{-3}$. Column (r) gives the approximation of $P_r(\text{delay})$ with reattempts, which involves numerically determining the solution of (22) and substituting the corrected parameters into (15).

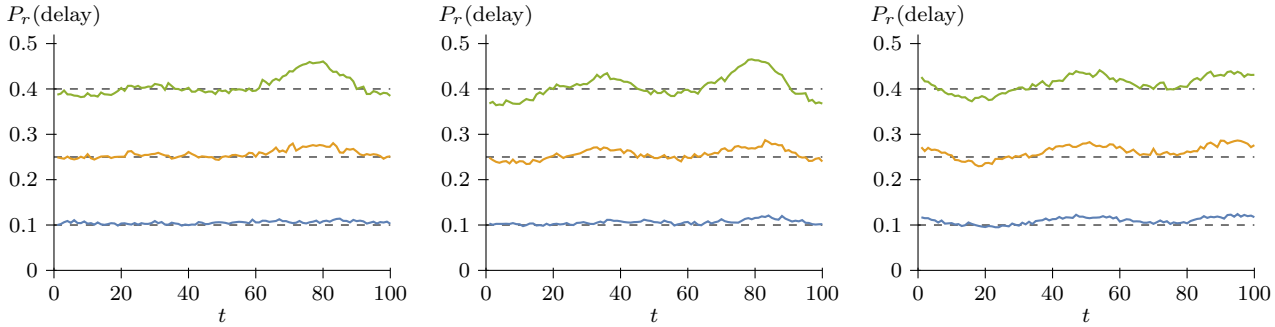
	$(\beta,\eta)=(1,1)$			$(\beta,\eta)=(0.5,0.5)$		
$\theta\backslash\delta$	0.01	1	100	0.01	1	100
10	0.101	0.123	0.186	0.201	0.250	0.359
1	0.171	0.181	0.200	0.360	0.386	0.415
0.1	0.211	0.209	0.212	0.469	0.471	0.476
0.01	0.221	0.219	0.219	0.496	0.495	0.495

Table 2: Simulation results on impact of δ.



(a) $\theta = 1$ (b) $\theta = 0.2$ (c) $\theta = 0.02$

Figure 5: Arrival rate function $R(t)$ (dashed) and staffing functions $s(t)$ (solid) and $n(t)$ (o) for different values of θ. The left vertical axis refers to $R(t)$ and $s(t)$, where the right axis refers to $n(t)$.



(a) $\theta = 1$ (b) $\theta = 0.2$ (c) $\theta = 0.02$

Figure 6: Simulated time-dependent delay probabilities in the cloud model with $\delta = 10^{-2}$, targets $\varepsilon = 0.1, \varepsilon = 0.25$ and $\varepsilon = 0.4$, and capacity levels determined by Algorithm 2.

We observe that, as R increases, the delay probabilities in the scenarios without and with reattempts converge to the limits $g_C(\beta, \eta)$ and $g_C(\beta - \alpha, \eta - \alpha/\sqrt{\theta})$, respectively, as expected. This supports the heuristic arguments that gave rise to the fixed point equation (22). We further note that the asymptotic QED approximations are accurate, even for relatively small systems.

4.2 Insensitivity for δ

We next investigate the impact of the mean reattempt time $1/\delta$. Table 2 displays simulation results for different settings of both δ and θ, with $R = 1000$ and two different pairs (β, η). Observe that the results hardly depend on δ and the differences are almost negligible for values of θ that are relevant for practical purposes. This is at first sight somewhat surprising, yet reassuring conclusion for both cloud user and provider. It can be understood as follows. For $\theta \ll \mu$, the vast majority of users in the system reside at the second infinite-server queue. Therefore, the system can roughly be regarded as an $M/G/n/n$ system, which in the QED regime means that users leave the system (VM positions become vacant) at rate $O(n)$. A reattempt time of length $1/n \approx \theta/R$ therefore already creates the desired leveling of workload.

4.3 Stabilizing performance

To illustrate Algorithm 2 we consider the time-varying load

$$R(t) = a + b \sin(2\pi t/T), \tag{28}$$

where we set $a = 1000$ as the mode, $b = 500$ as the amplitude and $T = 100$ as the cycle length. This system experiences large fluctuations in load volume over the course of one cycle. Since $\mu = 1$, this implies that one cycle on average consists of 100 service times at the host server queue. Due to relatively short service times with respect to the cycle length, the MOL approximation for the number of VM requests at the first queue is roughly equal to the original load, i.e. $R_1(t) \approx R(t)$. These short services at the host server compared to the cycle length are typical for VM provisioning, in which case the cycle is usually one day.

First, we examine the functions $s(t)$ and $n(t)$ as prescribed by (26)–(27) for $\theta = 1, 0.2, 0.02$ and $\varepsilon = 0.25$. The resulting values are depicted in Figure 5 together with the arrival rate function. Note that $n(t)$ lives on a different scale than $s(t)$, and has its own vertical axis at the right side of the plots.

For small and hence realistic values of θ, the function $n(t)$ displays a shifted phase compared to the real-time offered load, due to the the relatively long usage time of cloud users. The lag can be observed in (25). Hence, while the number of servers $s(t)$ allocated at time t are almost in phase with the arrival rate $R(t)$, $n(t)$ undergoes a shift of its peak capacity somewhat ahead in time. Observe also that $n(t)$ shows milder fluctuations when θ decreases. This can be attributed to the added hedge which is of order $\sqrt{R/\theta}$. Remark that the number of host servers on top of the strictly necessary amount, i.e. the hedge covering for the variability in the process, is relatively small. This illustrates the economies of scale that can be achieved in these large-scale systems. Next, we simulate the time-dependent process, given the staffing functions depicted in Figure 5, as well as the staffing functions designed for the target delay probabilities $\varepsilon = 0.1$ and

$\varepsilon = 0.4$ for the three values of θ. We use $\eta^* = 2$ as the input variable for Algorithm 2. The results of the simulations are depicted in Figure 6. In all cases, the time-dependent delay probability only mildly fluctuates around the target. As we increase ε, the stabilizing effect of the method weakens somewhat, which for other systems was also observed in [10].

5. CONCLUSION

We have designed an algorithm to let cloud system behavior stabilize around predefined target QoS-levels. While we have used the delay probability as an example, similar algorithms can be constructed for other performance measures, such as the mean delay or the blocking probability. The algorithm operates in the QED regime and is able to deal with time-varying loads by using the MOL method. It also allows blocked users to reattempt after exponential times, an easily implementable yet highly effective way of serving all users despite the inherently finite capacity of the system.

The dominance of the second phase ($\theta \ll \mu$) of the cloud model strongly impacts the performance of the algorithm. First, since most users reside in the second phase, the MOL method makes that in respond to a time-varying load $R(t)$, the $n(t)$-levels prescribed by the algorithm show a considerable lag and dampening effect. This is in sharp contrast with the prescribed $s(t)$-levels, which follow the movements in $R(t)$ almost instantaneously. Hence, in the prototypical cloud regime $\theta \ll \mu$ the number of host servers should adapt quicker to changes in the load than the number of VMs. Second, while for the $M/M/s/n$ system the reattempt rate should be taken relatively small, the cloud model is fairly insensitive to the precise choice of δ, again due to the dominance of the second phase. Hence, quick reattempts already have considerable impact, which is advantageous for the cloud users.

Apart from this difference in responsiveness, both $s(t)$ and $n(t)$ crucially influence the system performance, and need to be considered simultaneously to understand and influence the system behavior. The focus in the paper was on performance evaluation, but one could also formulate and optimize objective functions that quantify the trade-off between capacity costs and user dissatisfaction, including costs associated with each unit of s and n, costs charged for delay, and penalties for each blocked user. This gives rise to a new class of two-dimensional optimization problems.

Finally, let us mention that our algorithm is *proactive* [16], because capacity decisions are prescribed at the start of the planning period, and the offered load $R(t)$ is predicted based on the most recent information. As demand is realized over the course of a period, the cloud providers may want to deviate from this prescribed plan, as it detects significant differences between predicted and observed workload. This provisioning technique is called *reactive*. With the proactive provisioning algorithm developed in this paper serving as a benchmark, such online methods, see e.g. [18, 12], can be used as refinements for highly dynamic environments.

Both the two-dimensional optimization problems and the extension of our framework to more reactive algorithms present exciting directions for future research.

6. REFERENCES

[1] Amazon auto-scaling. https://aws.amazon.com/autoscaling.

[2] M. Armbrust, A. Fox, R. Griffith, A.D.. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. A view of cloud computing. *Commun. ACM*, 53(4):50–58, April 2010.

[3] J. Artelejo and A. Gómez-Corral. *Retrial Queueing Systems*. Springer-Verlag, 2008.

[4] S.C. Borst, A. Mandelbaum, and M.I. Reiman. Dimensioning large call centers. *Operations Research*, 52(1):17–34, January-February 2004.

[5] R.N. Calheiros, R. Ranjan, and R. Buyya. Virtual machine provisioning based on analytical performance and QoS in cloud computing environments. In *International Conference on Parallel Processing*, 2011.

[6] S.G. Eick, W.A. Massey, and W. Whitt. The physics of the $M_t/G/\infty$ queue. *Operations Research*, 41:731–742, 1993.

[7] G. Falin and J. Templeton. *Retrial Queues*. Chapman & Hall, 1997.

[8] S. Halfin and W. Whitt. Heavy-traffic limits for queues with many exponential servers. *Operations Research*, 29:567–588, 1981.

[9] O. Jennings and F. de Véricourt. Dimensioning large-scale memberschip services. *Operations Research*, 55(1):173–187, 2008.

[10] O.B. Jennings, A. Mandelbaum, W.A. Massey, and W. Whitt. Server staffing to meet time-varying demand. *Manag. Sci.*, 42(10):1383–1394, 1996.

[11] P. Khudyakov. Designing a call center with an IVR (Interactive Voice Response). Master's thesis, Technion, 2006.

[12] H.C. Lim, S. Babu, J.S. Chase, and S.S. Parekh. Automated control in cloud computing: challenges and opportunities. In *ACDCâĂŹ09*, 2009.

[13] Z. Liu, A. Wierman, Y. Chen, B. Razon, and N. Chen. Data center demand response: Avoiding the coincident peak via workload shifting and local generation. *Performance Evaluation*, 70:770–791, 2013.

[14] W.A. Massey and R.B. Wallace. An asymptotically optimal design of the $M/M/c/k$ queue. *Unpublished report*, 2004.

[15] P. Mell and T. Grance. The NIST definition of cloud computing. Technical report, NIST, 2011.

[16] O. Shoaib, Y. amd Das. Performance-oriented cloud provisioning: Taxonomy and survey. 2014.

[17] J. Tan, H. Feng, X. Meng, and L. Zhang. Heavy-traffic analysis of cloud provisioning. In *Proceedings of the 24th International Teletraffic Congress*, 2012.

[18] B. Urgaonkar, P. Shenoy, A. Chandra, and P. Goyal. Dynamic provisioning of multi-tier internet applications. In *Second International Conference on Autonomic Computing*, 2005.

[19] J.S.H. van Leeuwaarden, B.W.J. Mathijsen, and F. Sloothaak. Delayed workload shifting in many-server systems. *SIGMETRICS Perform. Eval. Rev.*, 43(2):10–12, September 2015.

[20] A. Wierman, Z. Liu, I. Liu, and H. Mohsenian-Rad. Opportunities and challenges for data center demand response. In *IEEE IGCC*, 2014.

[21] G. Yom-Tov. *Queues in hospitals: Queueing networks with Re-entering customers in the QED regime*. PhD thesis, Technion, 2010.