Department of Computer Science
George Mason University Technical Reports

4400 University Drive MS#4A5
Fairfax, VA 22030-4444 USA
http://cs.gmu.edu/ 703-993-1530

Model Driven Elasticity Control for Multi-Server Queues Under Traffic Surges in Cloud Environments

Venkat Tadakamalla
vtadakam@gmu.edu

Daniel A. Menascé
menasce@gmu.edu

Technical Report GMU-CS-TR-2018-1

Abstract

Many computer systems, such as Internet datacenters and cloud computing environments, consist of a multitude of servers that process user requests. Incoming requests that find all servers busy have to wait until a server becomes idle. This type of queuing system is known as a G/G/c system and has been extensively studied in the queuing literature under *steady state* conditions. This paper studies multi-server systems subject to *workload surges* of generic trapezoidal shapes during which the average arrival rate of requests exceeds the system's capacity. This paper's main contributions are: (1) Generic equations for surges of any shape. (2) A set of equations to estimate the impact of trapezoidal and triangular shaped surges on response time. (3) The design, implementation, and extensive evaluation of an autonomic controller for cloud server elasticity using a G/G/c simulator previously developed by the authors. The results show that our equations estimate with great accuracy the impact of surges on response time and that our autonomic controllers are able to successfully vary the number of servers to mitigate the impact of workload surges.

1 Introduction

Many computer systems, such as Internet datacenters and cloud computing environments, consist of a multitude of servers that process user requests. This paper addresses the case in which arriving requests select one out of c servers (e.g., web sites with multiple web servers at the front tier). If all c servers are busy, arriving requests wait in a waiting line until a server becomes available. We assume that servers are identical, i.e., they have the same processing capacity and can process any type of incoming request. All servers use a FCFS queuing discipline. When a server becomes idle, it takes the request at the front of the waiting line.

This type of queuing system is known as a G/G/c system in Kendall's notation [1]. In this notation, the first letter represents the type of distribution of the interarrival time of requests, the second letter indicates the distribution of the service time of requests, and c denotes the number of servers. The letter G stands for a generic distribution, while M (for Markovian or memoryless) stands for an exponential distribution, and D for a deterministic distribution (i.e., a constant value). These queuing systems have been extensively studied in *steady state*, i.e., when the average arrival rate of requests is smaller than the maximum rate at which the system can perform work, i.e., the *system capacity* (see e.g., [1, 2]). The ratio between the average arrival rate of requests and the system's capacity is called *traffic intensity* and is typically denoted by ρ in the queuing literature. A queuing system is in steady-state when $\rho < 1$. For some queuing systems (e.g., M/G/1, M/M/c) there are exact steady state results while for others (e.g., G/G/1 and G/G/c) there are approximations and/or bounds. Nevertheless, these exact or approximate results apply only to systems in steady state.

In a recent paper [3], we addressed the situation when a system is subject to *workload surges* (aka flash crowds), i.e, periods during which the arrival rate exceeds the system's capacity (see e.g., [4, 5, 6, 7, 8, 9, 10]). When that happens, the queue length grows continuously and so does the response time of requests. Additionally, the response time continues to increase even after the surge is over. In other words, the response time does not return to its steady state value as soon as the surge is over. In our preliminary work, we considered rectangle-shaped workload surges, i.e., the traffic intensity rises instantaneously, stays at a high value $\rho > 1$ for some time, and returns to its original value instantaneously. In [3], we derived analytical expressions for the effects of rectangular surges. In this paper, we extend that analysis to surges of any shape and use these results to study

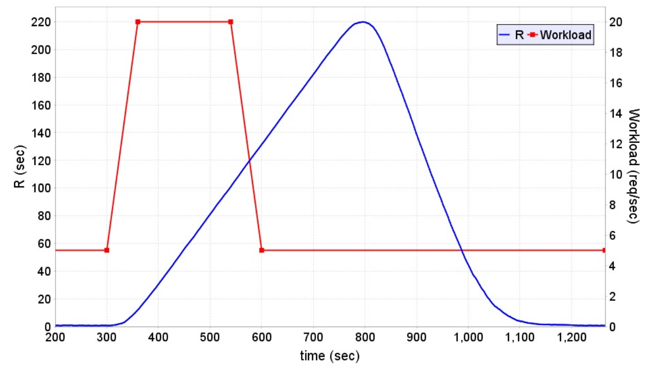

Figure 1: Example of trapezoidal workload surge and corresponding effect on the response time for 5 servers. Workload surge duration: 5 minutes; average service time: 0.5 sec; average arrival rate before and after surge: 5.0 requests/sec; average arrival rate during surge: 20.0 requests/sec; coefficient of variation of the interarrival time: 2; coefficient of variation of the service time: 2.

trapezoidal-shaped surges in detail. Our trapezoidal surge results yield rectangular and triangular surges as special cases.

As an illustration, consider Fig. 1 that shows a trapezoidal workload surge that lasts from $t = 300$ sec to $t = 600$ sec. The left axis shows the response time R and the right axis shows the average arrival rate. The workload intensity shows a surge from 5 to 20 requests/sec that lasts for 5 minutes, gradually increasing between $t = 300$ sec to $t = 360$ sec and gradually decreasing from $t = 540$ sec to $t = 600$ sec. The response time curve (blue curve) shows the response time of transactions that leave the system at a given time instant. As we can see, even though the traffic intensity returned to its steady-state value of 0.5 at time 600 sec, the response time peak of 220 seconds was observed at time 800 sec and it only returned to its pre-surge level at time 1,130 sec.

As illustrated above, workload surges generate very high response times that can be orders of magnitude higher than corresponding steady state values and can be very disruptive to users and damaging to organizations that provide computing services. Fluid approximations to queuing theory have been suggested as a way to analyze the transient behavior of queues [11]. In that formulation, customers arrive as a continuous fluid with a time-varying arrival rate. The equations we derive here have a fluid approximation flavor but go beyond what has been proposed previously.

Cloud providers, such as Infrastructure as a Service (IaaS), allow for resources in the form of virtual machines to be dynamically added or removed from the set of available resources to cope with traffic intensity variability so as to help ensure that response times stay within expected values. This is called elasticity (e.g., Amazon's Elastic Compute Cloud, EC2) and has been defined as *the degree to which a system is able to adapt to workload changes by provisioning and de-provisioning resources in an autonomic manner, such that at each point in time the available resources match the current demand as closely as possible* [12].

Many current approaches to elasticity are reactive, i.e., more capacity is added when performance degrades. However, these approaches suffer from the following drawbacks: (1) Deployment of extra capacity (e.g., more virtual machines) is not instantaneous, hence users will experience degraded performance while additional resources are not deployed. (2) There is a risk of overprovisioning or underprovisioning resources because it is not straightforward to determine how many resources have to be added or released as the traffic intensity varies.

Therefore, there is a need to control the system's capacity in an autonomic manner [13] so that the capacity can be dynamically changed in order to mitigate the effects of workload surges. The main contributions of this paper are: (1) The derivation of a generic property for surges of any shape. (2) The derivation of a set of equations to estimate the impact of trapezoidal and triangular shaped surges on response time. (3) The design, implementation, and extensive evaluation of an autonomic controller for cloud server elasticity using a G/G/c simulator we developed for [3].

The rest of this paper is organized as follows. Section 2 presents core results including notation, the system model, the surge model, assumptions, and two generic results for surges of any shape. Section 3 presents a set of analytical expressions that can be used to estimate the effects of trapezoidal shaped workload surges. Analytical expressions for special cases of trapezoidal shaped workload surges (i.e., rectangular and triangular) are derived in Section 4. Section 5 presents the validations of the analytical expression derived using simulation experiments. Section 6 presents the algorithms used by two autonomic controllers to dynamically control the elasticity of a multi-server system. This autonomic controller is evaluated in Section 7. Related work is presented in Section 8. Finally, Section 9 concludes the paper and discusses future work.

2 Core Results

This section describes the system model and its notation, the surge model and its notation, the assumptions about the surge, and two generic results about the surge.

2.1 System Model

A multi-server queue is modeled as a G/G/c queuing system (see Fig. 2) and is characterized by the following parameters:

- c: number of servers

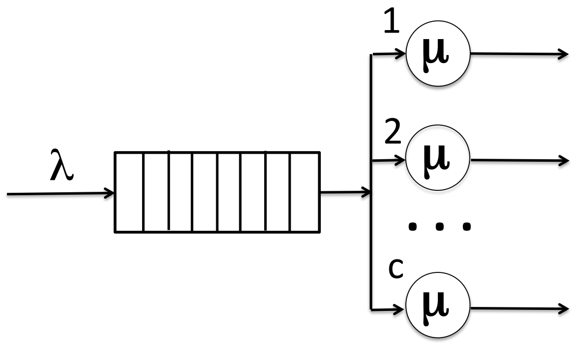

Figure 2: Diagram of a G/G/c queue

- λ: average arrival rate of requests. Thus, the average interarrival time is $1/\lambda$.
- μ: average service rate of each of the c servers. Thus, the average service time of a request is $1/\mu$.
- C_a: coefficient of variation of the interrarival time, i.e., the ratio between the standard deviation of the interrarival time and its average.
- C_s: coefficient of variation of the service time, i.e., the ratio between the standard deviation of the service time and its average.
- ρ: traffic intensity. $\rho = \lambda/(\mu c)$. For the system to be stable, ρ has to be less than 1.
- $R(t)$: response time estimate for requests that leave the system at time t.
- $n_q(t)$: average queue length (i.e., number of requests in the waiting line plus number of requests being served) at time t.

As mentioned above, there are no exact solutions for the average response time or average queue length for the G/G/c queue. Most approximations for steady state are based on the first two moments of the interarrival time and service time distributions (i.e., are a function of C_a and C_s) [14]. Our simulation experiments show results for different values of C_a and C_s.

2.2 Surge Model

Figure 3 shows the variation of the traffic intensity $\rho(t) = \lambda(t)/(\mu c)$ as a function of time t for a workload surge that starts at $t = t_{\text{beg}}$ and ends at $t = t_{\text{end}}$. The green horizontal line corresponds to $\rho(t) = 1$, which means that $\lambda(t) = \mu c$. The portion of the $\rho(t)$ curve between t_a and t_b represents the portion of the surge when the system is *unstable*, i.e., $\rho(t) > 1$.

The notation used for the surge model is the following:

- t_{beg}: time at which the surge begins.
- t_{end}: time at which the surge ends.
- t_a: start time of the unstable period of the surge.
- t_b: end time of the unstable period of the surge.
- ρ_1: pre-surge traffic intensity.
- ρ_2: maximum value of $\rho(t)$ achieved during the surge.
- R_1: average pre-surge response time.
- R_2: average peak response time caused by the surge.
- t_{norm}: arrival time of the first request that sees the response time return to its pre-surge value, i.e., $R(t'_{\text{norm}}) = R_1$ where t'_{norm} is the departure time of such request.
- Ψ_1: net number of accumulated requests in the interval (t_a, t_b).
- Ψ_2: net number of accumulated requests in the interval (t_b, t_{end}).
- Ψ_3: net number of accumulated requests in the interval $(t_{\text{end}}, t_{\text{norm}})$.

The number of requests that arrive during (t_a, t_b) is $\int_{t_a}^{t_b} \lambda(t)dt$. During this interval, according to the heavy load assumption ASM-1 below, the rate at which requests depart is μc and therefore the number of requests that depart in the interval (t_a, t_b) is $\mu c(t_b - t_a)$. Therefore, the net number of accumulated requests during (t_a, t_b) is

$$\begin{aligned}
\Psi_1 &= \left[\int_{t_a}^{t_b} \lambda(t)dt\right] - \mu c(t_b - t_a) \\
&= \left[\int_{t_a}^{t_b} \mu c\, \rho(t)dt\right] - \mu c(t_b - t_a) \\
&= \mu c\left[\int_{t_a}^{t_b} \rho(t)dt - (t_b - t_a)\right] \\
&= \mu c\left[\int_{t_a}^{t_b} (\rho(t) - 1)dt\right]. \qquad (1)
\end{aligned}$$

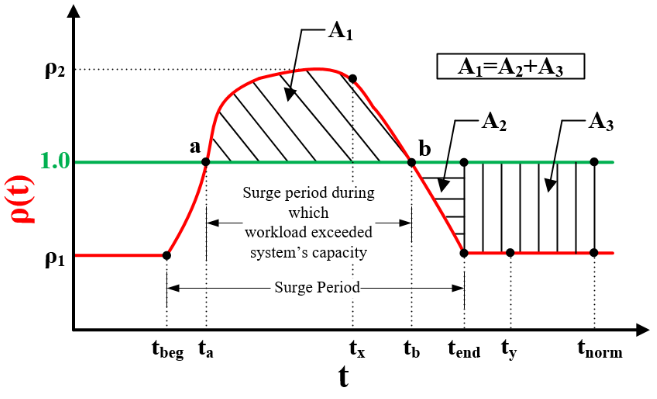

Figure 3: Surge depicted as $\rho(t)$ vs. t

The integral in Eq. 1 is the area marked as A_1 in Fig. 3.

Using a similar reasoning as above and assuming, according to the traffic jam effect assumption (ASM-2 below), that all c servers will be busy during the interval (t_b, t_{end}), we have

$$\begin{aligned}\Psi_2 &= \left[\int_{t_b}^{t_{\text{end}}} \lambda(t)dt\right] - \mu c(t_{\text{end}} - t_b) \\ &= \left[\int_{t_b}^{t_{\text{end}}} \mu c\, \rho(t)dt\right] - \mu c(t_{\text{end}} - t_b) \\ &= \mu c\left[\int_{t_b}^{t_{\text{end}}} (\rho(t)-1)dt\right]. \qquad (2)\end{aligned}$$

The integral $\int_{t_b}^{t_{\text{end}}}(\rho(t)-1)dt$ is negative because $\rho(t) < 1$ in the interval (t_b, t_{end}). So, the net number of accumulated requests in this interval is negative, i.e., more requests leave than arrive. Thus, $\Psi_2 < 0$.

Similarly to what we did above, we can write

$$\Psi_3 = \mu c \int_{t_{\text{end}}}^{t_{\text{norm}}} (\rho(t)-1)dt. \qquad (3)$$

But, because $\rho(t) = \rho_1$ in the interval $(t_{\text{end}}, t_{\text{norm}})$, we have

$$\Psi_3 = \mu c\, (\rho_1 - 1)(t_{\text{norm}} - t_{\text{end}}). \qquad (4)$$

Note that $\Psi_3 < 0$ because $\rho_1 < 1$ in the interval $(t_{\text{end}}, t_{\text{norm}})$.

The value of t_{norm} can be obtained by observing that the requests added by the surge during t_a and t_b have to be removed from the system between t_b and t_{norm}. So, $\Psi_1 + \Psi_2 + \Psi_3 = 0$. Because Ψ_3 depends on t_{norm} one can easily compute t_{norm} by combining Eqs. 1, 2 , and 4. However, when $A_1 =\mid A_2 \mid$, $A_3 = 0$ and $t_{\text{norm}} = t_{\text{end}}$. So,

$$t_{\text{norm}} = t_{\text{end}} + max\left(0, \frac{\int_{t_a}^{t_{\text{end}}}(\rho(t)-1)dt}{1-\rho_1}\right). \qquad (5)$$

2.3 Assumptions

We use the following assumptions:

- ASM-1 (*heavy load assumption*): When the traffic intensity $\rho(t)$ exceeds 1, all c servers will be busy 100% of the time and the system throughput will be equal to μc.
- ASM-2 (*traffic jam effect*): All c servers will continue to be 100% busy until some time t_{norm} after the end of the surge. So, the system throughput will be approximated as μc until $t = t_{\text{norm}}$.
- ASM-3: (*fast rampup*): The traffic intensity increases very fast between t_{beg} and t_a. More precisely, $(1-\rho_1)/(t_a - t_{\text{beg}}) \gg 1/(\text{time unit})$. Therefore, there is not enough time for the queue to increase significantly between t_{beg} and t_a. Hence, $n_q(t_a) \approx n_q(t_{\text{beq}})$.

2.4 Generic Property About a Surge

We now show the following useful generic surge property:

Generic Surge Property: Consider the *heavy load* assumption that $\rho(t) \gg 1$ during a significant portion of the interval $[t_a, t_b]$ (see Fig. 3) and consider a tagged request that arrives at time t_x for $t_a \leq t_x \leq t_b$. Then, the tagged request leaves the system at time at time $t_x^{'}$ given by

$$t_x^{'} \approx t_x + \int_{t=t_a}^{t=t_x}[\rho(t)-1]dt \qquad (6)$$

and the response time $R(t_x^{'})$ is given by

$$R(t_x^{'}) \approx \int_{t=t_a}^{t=t_x}[\rho(t)-1]dt. \qquad (7)$$

Proof: Similarly to our derivations of Ψ_1, we define Ψ_x as the net number of requests accumulated in the interval (t_a, t_x). So,

$$\Psi_x = \mu c \int_{t_a}^{t_x}(\rho(t)-1)dt. \qquad (8)$$

Thus, the average number of requests in the system at time t_x is

$$n_q(t_x) = n_q(t_a) + \Psi_x. \qquad (9)$$

The tagged request will depart at time $t_x^{'}$ after all $n_q(t_x)$ requests it finds in the system when it arrives are served and after the tagged request itself is served, which takes $1/\mu$ time units. Due to the heavy traffic assumption (ASM-1), the throughput in the interval (t_a, t_b) is considered to be equal to μc. Thus, it will take $n_q(t_x)/(\mu c)$ time units to serve all $n_q(t_x)$ requests. So,

$$\begin{aligned}t_x^{'} &= t_x + \frac{n_q(t_x)}{\mu c} + \frac{1}{\mu} = t_x + \frac{n_q(t_a)+\Psi_x}{\mu c} + \frac{1}{\mu} \\ &= t_x + \frac{n_q(t_a)}{\mu c} + \int_{t_a}^{t_x}(\rho(t)-1)dt + \frac{1}{\mu} \qquad (10)\end{aligned}$$

But, according to the fast rampup assumption (ASM-3), $n_q(t_{\text{beq}}) \approx n_q(t_a)$ and Eq. 10 becomes

$$t_x^{'} = t_x + \frac{n_q(t_{\text{beg}})}{\mu c} + \frac{1}{\mu} + \int_{t_a}^{t_x}(\rho(t)-1)dt. \quad (11)$$

But,

$$\frac{n_q(t_{\text{beg}})}{\mu c} + \frac{1}{\mu} \leq \frac{n_q(t_{\text{beg}})}{\rho_1 \mu c} + \frac{1}{\mu} = \frac{n_q(t_{\text{beg}})}{\lambda_1} + \frac{1}{\mu} = R_1 \qquad (12)$$

Thus,

$$t_x^{'} \approx t_x + R_1 + \int_{t_a}^{t_x}(\rho(t)-1)dt. \qquad (13)$$

By definition, the response time of a request is the difference between the time it departs and the time it arrives. So,

$$R(t_x^{'}) = t_x^{'} - t_x \approx R_1 + \int_{t=t_a}^{t=t_x}[\rho(t)-1)]dt. \qquad (14)$$

If the pre-surge response time $R_1 \ll \int_{t=t_a}^{t=t_x}[\rho(t)-1)]dt$, i.e., t_x is much closer to t_b than to t_a, the above equations can be further approximated as

$$t_x^{'} \approx t_x + \int_{t=t_a}^{t=t_x}[\rho(t)-1)]dt \quad (15)$$

and

$$R(t_x^{'}) \approx \int_{t=t_a}^{t=t_x}[\rho(t)-1)]dt. \quad (16)$$

3 Trapezoidal Shaped Workload Surges

We now consider for the rest of this paper that the traffic surge is shaped as a trapezoid, which is a very good approximation for generic surges as the one depicted in Fig. 3.

Figure 4 shows a trapezoidal workload surge; the top curve in that figure shows $\rho(t)$ and the bottom one shows $R(t)$, defined as the response time of requests leaving the system at time t.

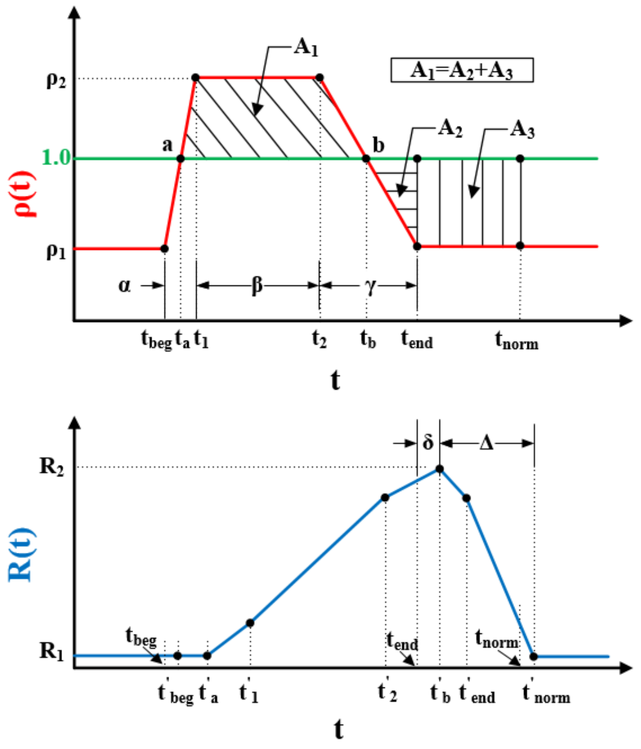

Figure 4: Trapezoidal workload surge. Top: variation of the traffic intensity $\rho(t)$ vs. time. Bottom: variation of the response time $R(t)$ vs. time.

A trapezoidal surge has three phases: a *ramp-up* phase in which the average arrival rate increases until it peaks, a *peak* phase during which the average arrival rate remains at its peak value, and a *ramp-down* phase during which the average arrival rate decreases to the value it had before the start of the ramp-up phase. Let the average arrival rate before the beginning of the ramp-up phase be λ_1; therefore the traffic intensity is $\rho_1 = \lambda_1/(\mu c)$. The average arrival rate during the peak phase is denoted by λ_2 and therefore the traffic intensity during this phase is $\rho_2 = \lambda_2/(\mu c)$. During a portion α, $(t_a - t_{\text{beg}})$, of the ramp-up period, the traffic intensity has a value $\rho_1 < 1$ (i.e., the system is stable). Then, $\rho(t)$ increases to $\rho_2 > 1$ and stays at that level during the unstable portion of the surge period ($\beta = t_2 - t_1$). Then, the traffic intensity decreases back to ρ_1 during the ramp-down period of duration $\gamma = t_{\text{end}} - t_2$. Thus, the characteristics of the trapezoidal workload are determined by the following attributes: $t_{\text{beg}}, \alpha, \beta, \gamma, \rho_1$ and ρ_2.

The $\rho(t)$ curve in Fig. 4 can be written using the following piece-wise linear relationships:

$$\rho(t) = \begin{cases} \rho_1 & t \le t_{\text{beg}} \\ \rho_1 + \frac{\rho_2-\rho_1}{t_1-t_{\text{beg}}}(t-t_{\text{beg}}) & t_{\text{beg}} < t \le t_1 \\ \rho_2 & t_1 < t \le t_2 \\ \rho_2 + \frac{\rho_1-\rho_2}{t_{\text{end}}-t_2}(t-t_2) & t_2 < t \le t_{\text{end}} \\ \rho_1 & t_{\text{end}} < t. \end{cases} \quad (17)$$

By definition (see Fig. 4),

$$\begin{aligned} \alpha &= t_1 - t_{\text{beg}} \\ \beta &= t_2 - t_1 \\ \gamma &= t_{\text{end}} - t_2. \end{aligned} \quad (18)$$

For convenience we define the constant K as:

$$K = \frac{\rho_2 - 1}{\rho_2 - \rho_1}. \quad (19)$$

Using the above equations and Fig. 4, we can now express t_a and t_b as:

$$t_a = t_1 - K\alpha = t_{\text{beg}} + (1-K)\alpha \quad (20)$$
$$t_b = t_2 + K\gamma = t_{\text{end}} - (1-K)\gamma. \quad (21)$$

We derive in the following subsections analytic expressions for the following metrics:

- $t_b^{'}$: time at which the response time reaches its peak value R_2.
- $R_2 = R(t_b^{'})$: estimate for the peak response time.
- $\delta = t_b^{'} - t_{\text{end}}$: response time lag, i.e., estimated duration of the interval between the end of the ramp-down phase and the time when the response time peaks.
- $\Delta = t_{\text{norm}}^{'} - t_b^{'}$: estimated time needed for the response time to decrease from its peak value of R_2 to its value R_1 before the surge started.

We now compute the areas A_1 and A_2 for the trapezoidal surge using the geometry of Fig. 4. A_1 is the area of a trapezoid with bases $(t_b - t_a)$ and β and height $\rho_2 - 1$. Thus,

$$A_1 = \frac{\beta + t_b - t_a}{2}(\rho_2 - 1). \tag{22}$$

Replacing t_a and t_b from Eqs. (20) and (21) yields

$$A_1 = \left[\beta + \frac{K(\alpha+\gamma)}{2}\right](\rho_2 - 1). \tag{23}$$

A_2 is the area of a triangle with base $t_{\text{end}} - t_b$ and height $1 - \rho_1$. Thus,

$$A_2 = \frac{(t_{\text{end}} - t_b)(1-\rho_1)}{2}. \tag{24}$$

But, according to Eq. 21, $t_{\text{end}} - t_b = (1-K)\gamma$. So, A_2 can be rewritten as

$$A_2 = \frac{(1-K)\gamma(1-\rho_1)}{2}. \tag{25}$$

We can now compute t_{norm} for the trapezoidal surge case using Eq. 5 as

$$\begin{aligned} t_{\text{norm}} &= t_{\text{end}} + max\left(0, \frac{\int_{t_a}^{t_{\text{end}}}(\rho(t)-1)dt}{1-\rho_1}\right) \\ &= t_{\text{end}} + max\left(0, \frac{A_1 - A_2}{1-\rho_1}\right). \end{aligned} \tag{26}$$

3.1 Estimating $t_b^{'}$ and R_2

We use the property of a generic surge derived in the previous section to derive R_2. The peak response time will be seen by a request that arrives at time t_b, i.e., right at the end of the unstable portion of the surge. This request leaves the system at time $t_b^{'}$. Thus, according to Eqs. (6) and (7), we have

$$t_b^{'} \approx t_b + \int_{t=t_a}^{t=t_b}[\rho(t)-1)]dt \tag{27}$$

and the response time $R_2 = R(t_b^{'})$ is given by

$$R_2 \approx \int_{t=t_a}^{t=t_b}[\rho(t)-1)]dt. \tag{28}$$

Thus,

$$t_b^{'} = t_b + A_1 \tag{29}$$

and

$$R_2 = A_1. \tag{30}$$

3.2 Estimating δ

From Fig. 4, $\delta = t_b^{'} - t_{\text{end}}$, the estimated duration of the interval between the end of the ramp-down phase and the time when the response time peaks. From Eqs. (29) and (21) we get

$$\delta = t_b^{'} - t_{\text{end}} = t_b + A_1 - (t_b + (1-K)\gamma) = A_1 - (1-K)\gamma \tag{31}$$

3.3 Estimating Δ

From Fig. 4, we can see that

$$\Delta = t_{\text{norm}}^{'} - t_b^{'} \tag{32}$$

But, because a request that arrives at time t_{norm} has an average response time of R_1, we have that $t_{\text{norm}}^{'} = t_{\text{norm}} + R_1$. Then,

$$\Delta = t_{\text{norm}} + R_1 - t_b^{'} \tag{33}$$

Using Eqs. 26 and 29 in Eq. 33 yields

$$\Delta = (1-K)\gamma + R_1 + \frac{\rho_1 A_1 - A_2}{1-\rho_1} \tag{34}$$

where A_1 and A_2 are given by Eqs. 23 and 25, respectively. If R_1 is much smaller than the other terms in the equation, which is certainly the case for heavy load conditions, then

$$\Delta \approx (1-K)\gamma + \frac{\rho_1 A_1 - A_2}{1-\rho_1}. \tag{35}$$

3.4 Estimating Points on the Response Time Curve

The bottom of Fig. 4 shows the response time curve $R(t)$ that results from a trapezoidal surge. The figure shows that the shape of this curve can be estimated by seven important points along the time axis. Remember that we use the notation $t^{'}$ as the departure instant of a request that arrives at time t. So, $R(t^{'}) = t^{'} - t$. Table 1 shows x and y coordinates of these points and explains the reason for the expression for the value of $R(t)$. Note that for all points within the surge we use the general surge property shown in Eq. 7 to compute the value of the response time. The integral in Eq. 7 is easily computed from the geometric characteristics of the trapezoidal surge shown in the top graph of Fig. 4. Points 1 and 7 are points that represent steady-state before and after the surge.

4 Special Cases of the Trapezoidal Workload Surge

We discuss here two special cases of the trapezoidal surge: rectangular and triangular surges.

4.1 Rectangular Workload Surge

We can derive equations for rectangular surges as a special case of trapezoidal ones. Doing so yields the same results obtained in our previous paper [3].

A rectangular surge is characterized by $t_1 = t_{\text{beg}}$ and $t_{\text{end}} = t_2$. Thus, $\alpha = \gamma = 0$. Therefore, we get the following values for R_2, δ, and Δ using Eqs. 30, 31, and 35.

$$R_2 = (\rho_2 - 1)\beta$$

Table 1: Salient points on response time curve for trapezoidal shaped workload surges

i	x_i	y_i	Explanation
1	t'_{beg}	R_1	Steady-state pre-surge
2	$t'_a \approx t_a + R_1$	$\approx R_1$	Assump. ASM-3
3	$t'_1 = t_1 + R(t'_1)$	$R(t'_1) \approx R_1 + (\frac{K\alpha}{2})(\rho_2 - 1)$	Eq. 14
4	$t'_2 = t_2 + R(t'_2)$	$R(t'_2) \approx R_1 + (\beta + \frac{K\alpha}{2})(\rho_2 - 1)$	Eq. 14
5	$t'_b = t_b + R(t'_b)$	$R(t'_b) \approx R_1 + A_1 = R_2$	Eq. 14
6	$t'_{end} = t_{end} + R(t'_{end})$	$R(t'_{end}) \approx R_1 + A_1 - A_2$	Eq. 14
7	$t'_{norm} = t_{norm} + R_1$	R_1	Steady-state post surge

$$\begin{aligned} \delta &= (\rho_2 - 1)\beta \\ \Delta &= \frac{\rho_1(\rho_2 - 1)}{1 - \rho_1}\beta \end{aligned} \tag{36}$$

The results above match exactly those obtained in our previous paper [3].

4.2 Triangular Workload Surge

We present here results for three types of triangular workload surges (see Fig.5), derived from the trapezoidal surge in Fig. 4 by making $\beta = 0$, i.e., $(t_2 = t_1)$. For all three cases, the area A_1 is given by

$$A_1 = \frac{(\rho_2 - 1)(t_b - t_a)}{2}. \tag{37}$$

4.2.1 Triangular Workload Surge, Type 1 ($\alpha > 0, \gamma = 0, \Rightarrow A_2 = 0$)

For this shape of workload surge, we have:

$$\begin{aligned} R_2 &= A_1 = (t_b - t_a)(\rho_2 - 1)/2 \\ \delta &= A_1 = (t_b - t_a)(\rho_2 - 1)/2 \\ \Delta &= \frac{\rho_1 A_1}{1 - \rho_1} = \frac{\rho_1(\rho_2 - 1)(t_b - t_a)}{2(1 - \rho_1)} \end{aligned} \tag{38}$$

where $t_b - t_a = K\alpha$(see top of Fig. 5).

4.2.2 Triangular Workload Surge, Type 2 ($\alpha = 0,\ \gamma > 0$)

For this workload surge, we have:

$$\begin{aligned} R_2 &= A_1 = (t_b - t_a)(\rho_2 - 1)/2 \\ \delta &= A_1 - (1 - K)\gamma = (t_b - t_a)(\rho_2 - 1)/2 - (1 - K)\gamma \\ \Delta &= (1 - K)\gamma + \frac{\rho_1 A_1 - A_2}{1 - \rho_1} \end{aligned} \tag{39}$$

where $t_b - t_a = K\alpha$ and $A_2 = (1 - \rho_1)(t_{\text{end}} - t_b)/2$.

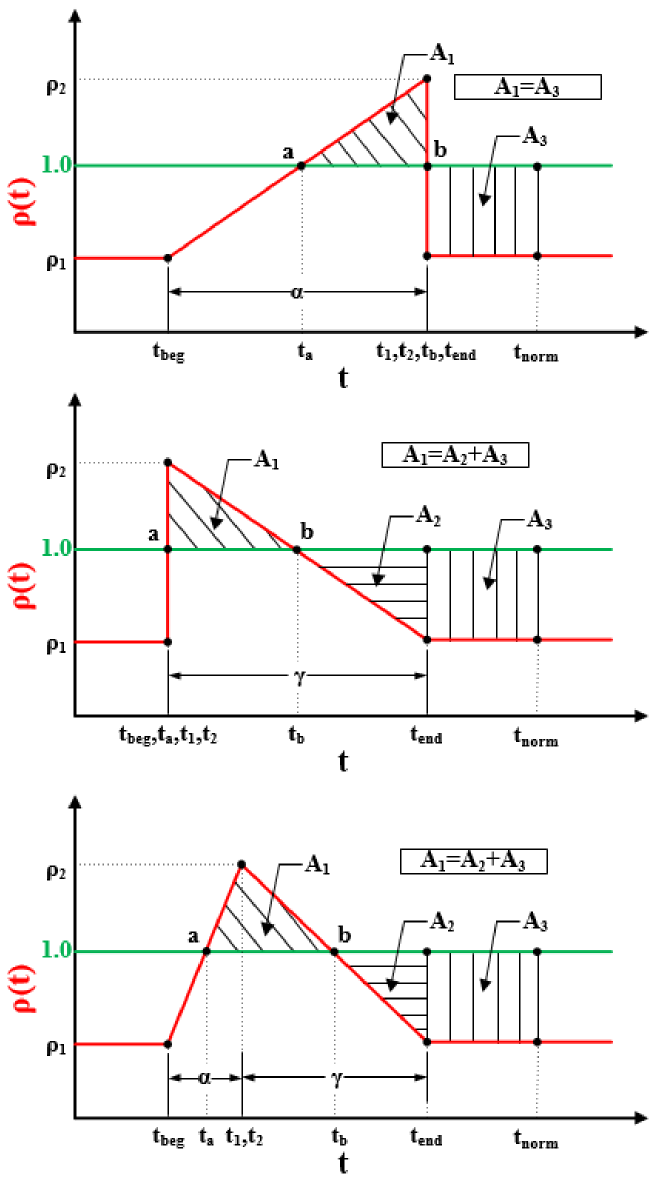

Figure 5: Triangular workload surge - variation of the traffic intensity. Top: Type 1. Middle: Type 2. Bottom: Type3

4.2.3 Triangular Workload Surge, Type 3 ($\alpha > 0, \gamma > 0$)

For this shape of workload surge, we have:

$$R_2 = A_1 = (t_b - t_a)(\rho_2 - 1)/2$$

$$\begin{aligned}\delta &= A_1 - (1-K)\gamma = (t_b - t_a)(\rho_2 - 1)/2 - (1-K)\gamma \\ \Delta &= (1-K)\gamma + \frac{\rho_1 A_1 - A_2}{1-\rho_1} \qquad (40)\end{aligned}$$

where $t_b - t_a = K(\alpha + \gamma)$ and $A_2 = (1-\rho_1)(t_{\text{end}} - t_b)/2$.

Therefore, all three types of triangular workload surges have the same value of R_2 and types 2 and 3 have the same expression for δ and Δ.

5 Validations

We validated the equations derived in the previous section with a G/G/c simulator we developed previously [3]. This simulator also includes the autonomic elasticity controller described in Section 6.

5.1 Comparing Estimates with Simulation for Trapezoidal Surges

Figure 6 shows a surge lasting 300 seconds during the interval (300 sec, 600 sec) and the resulting response time curve for a G/G/5 system. The following parameters were used to generate this figure: $\lambda_1 = 5$ req/sec; $\lambda_2 = 20$ req/sec; $\mu = 2$ req/sec; $C_a = C_s = 2.0$; $\alpha = 60$ sec; $\beta = 180$ sec; $\gamma = 60$ sec. Therefore, $\rho_1 = 0.5$, $\rho_2 = 2$, $K = (2-1)/(2-0.5) = 2/3$, $t_a = 320$ sec, $t_b = 580$ sec, $A_1 = 220$ sec, and $A_2 = 5$ sec. A total of 1.5 million requests were processed by the G/G/5 simulator for 100 independent runs.

The figure shows a peak response time of about 220 sec, while the estimated value of R_2, according to Eq. (22), is $(\rho_2 - 1)(\beta + t_b - t_a)/2 = (2-1) \times (180 + 580 - 320)/2 = 220$ sec. This is exactly the peak response time observed in the figure. This peak response time occurs at time 800 sec according to Fig. 6. Since the surge ends at time 600 sec, $\delta \approx$ 800 - 600 = 200 sec. This is the same value estimated by Eq. (31) for δ as $220 - (1 - 2/3) \times 60 = 200$ sec. Finally, the estimated value of Δ, according to Eq. (35), is $(1 - 2/3) \times 60 + (0.5 \times 220 - 5.0)/(1 - 0.5) = 230$ sec. This estimated value would take us to time 800 sec + 230 sec = 1,030 sec for t'_{norm}, which is close to the 1,090 sec shown in the figure.

Table 2 shows several other comparisons between estimated values and simulation results. The table illustrates the results of eight experiments with different sets of parameters. For each experiment, we report the value of the peak response time R_2, δ, and Δ obtained by the G/G/c simulator and through the estimates. The simulation values are averages over 100 runs with 95% confidence intervals. The number of requests processed in each set of 100 runs varied between one million and 1.7 million. We also report the percent error between the simulation and the analytic estimates as

$$\epsilon = 100 \times (\text{simulation} - \text{estimate})/\text{simulation}. \qquad (41)$$

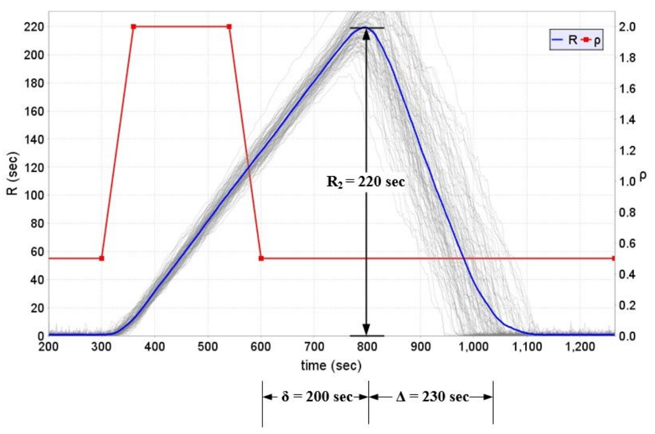

Figure 6: Example of trapezoidal workload surge and corresponding effect on the response time for $c = 5$ servers (average over 100 runs). Workload surge duration, $\alpha = 60$ sec, $\beta = 180$ sec, $\gamma = 60$ sec, $t_{beg} - t_{end} =$ 300 sec; average service time: 0.5 sec; average arrival rate before and after surge: 5 req/sec; average arrival rate during surge: 20 req/sec; coefficient of variation of the interarrival time: 2; coefficient of variation of the service time: 2

Experiments 1-6 in Table 2 are for G/G/5 and experiments 7 and 8 are for M/M/5 and D/D/5, respectively. The surge duration ($\alpha + \beta + \gamma = t_{end} - t_{beg}$) varies from experiment to experiment as indicated in the table. The pre-surge traffic intensity, ρ_1, was 0.6 in all cases and the maximum traffic intensity during the surge, ρ_2, was 1.5 in all cases except for experiment 6, when it was 2.0. The table also indicates for each experiment the values of the coefficients of variation C_a and C_s. Clearly, for M/M/5 $C_a = C_s = 1$ and for D/D/5 $C_a = C_s = 0$. As it can be seen, the percent error ϵ is very small (less than 9.9%) in all cases. The largest errors occur for δ, in which case the maximum error is 9.9% and occurs in experiment 1. Even in that case, the estimated value of δ (13.3 sec) is very close to both the mean value of (14.76 sec) or the lower bound (11.96 sec) of the 95% confidence interval. We also observe that as the surge duration $(\alpha + \beta + \gamma)$ increases from 180 sec to 420 sec from experiment 1 to 5, the error ϵ for δ tends to decrease. This is expected because of the heavy load assumption used in the derivation of the analytic estimates. For the same reason, higher values of ρ_2 improve the accuracy of the estimates. This can be seen in experiment 6 that uses $\rho_2 = 2.0$.

Figures 7, 8, and 9 illustrate the pictorial comparison of the workload surge estimates with those of simulation experiments for three different sets of trapezoidal shaped workload surges. Each set consists of $D/D/5$, $M/M/5$, and $G/G/5$ systems. In each of the figures, the cyan curve is the response time computed using the analytical estimates; the blue curve is the average response time of 100 independent simulation runs. The results clearly illustrate that the surge estimates are very close to those

Table 2: Comparison between estimates and simulation for Trapezoidal Surges

	[#1] $G/G/5$; $C_a = 1.4$; $C_s = 1.3$; $\rho_1 = 0.6$; $\rho_2 = 1.5$; $R_1 = 0.608$ sec; $\alpha = 45$ sec; $\beta = 60$ sec; $\gamma = 75$ sec		
	R_2 (sec)	δ (sec)	Δ (sec)
Simulation	49.08 ± 1.6	14.76 ± 2.8	92.10 ± 4.0
Estimated	46.70	13.30	86.70
ϵ	4.9%	9.9%	5.9%
	[#2] $G/G/5$; $C_a = 1.4$; $C_s = 1.3$; $\rho_1 = 0.6$; $\rho_2 = 1.5$; $R_1 = 0.608$ sec; $\alpha = 45$ sec; $\beta = 120$ sec; $\gamma = 75$ sec		
	R_2 (sec)	δ (sec)	Δ (sec)
Simulation	81.19 ± 1.8	46.64 ± 3.0	138.10 ± 4.5
Estimated	76.70	43.30	131.70
ϵ	5.5%	7.2%	4.6%
	[#3] $G/G/5$; $C_a = 1.4$; $C_s = 1.3$; $\rho_1 = 0.6$; $\rho_2 = 1.5$; $R_1 = 0.608$ sec; $\alpha = 45$ sec; $\beta = 180$ sec; $\gamma = 75$ sec		
	R_2 (sec)	δ (sec)	Δ (sec)
Simulation	111.70 ± 2.0	75.26 ± 3.3	186.12 ± 5.2
Estimated	106.70	73.30	176.70
ϵ	4.5%	2.6%	5.1%
	[#4] $G/G/5$; $C_a = 1.4$; $C_s = 1.3$; $\rho_1 = 0.6$; $\rho_2 = 1.5$; $R_1 = 0.608$ sec; $\alpha = 45$ sec; $\beta = 240$ sec; $\gamma = 75$ sec		
	R_2 (sec)	δ (sec)	Δ (sec)
Simulation	141.60 ± 2.4	107.72 ± 3.5	226.90 ± 5.5
Estimated	136.70	103.30	221.70
ϵ	3.5%	4.1%	2.3%
	[#5] $G/G/5$; $C_a = 1.4$; $C_s = 1.3$; $\rho_1 = 0.6$; $\rho_2 = 1.5$; $R_1 = 0.608$ sec; $\alpha = 45$ sec; $\beta = 300$ sec; $\gamma = 75$ sec		
	R_2 (sec)	δ (sec)	Δ (sec)
Simulation	171.84 ± 3.0	136.88 ± 3.9	276.24 ± 6.7
Estimated	166.70	133.30	266.70
ϵ	3.0%	2.6%	3.5%
	[#6] $G/G/5$; $C_a = 2.0$; $C_s = 2.0$; $\rho_1 = 0.5$; $\rho_2 = 2.0$; $R_1 = 0.633$ sec; $\alpha = 45$ sec; $\beta = 300$ sec; $\gamma = 75$ sec		
	R_2 (sec)	δ (sec)	Δ (sec)
Simulation	348.82 ± 4.9	322.38 ± 5.7	366.00 ± 8.1
Estimated	340.00	315.00	352.50
ϵ	2.5%	2.3%	3.7%
	[#7] $M/M/5$; $C_a = 1.0$; $C_s = 1.0$; $\rho_1 = 0.6$; $\rho_2 = 1.5$; $R_1 = 0.559$ sec; $\alpha = 45$ sec; $\beta = 300$ sec; $\gamma = 75$ sec		
	R_2 (sec)	δ (sec)	Δ (sec)
Simulation	170.43 ± 1.9	133.88 ± 2.8	276.48 ± 4.7
Estimated	166.70	133.30	266.70
ϵ	2.2%	0.4%	3.5%
	[#8] $D/D/5$; $C_a = 0.0$; $C_s = 0.0$; $\rho_1 = 0.6$; $\rho_2 = 1.5$; $R_1 = 0.500$ sec; $\alpha = 45$ sec; $\beta = 300$ sec; $\gamma = 75$ sec		
	R_2 (sec)	δ (sec)	Δ (sec)
Simulation	167.26 ± 0.0	133.00 ± 0.0	270.00 ± 0.0
Estimated	166.70	133.30	266.70
ϵ	0.3%	-0.2%	1.2%

of simulation experiments.

5.2 Analyzing the Effects of C_a and C_s for Trapezoidal Surges

We now examine the effects of C_a and C_s on the three metrics we proposed in Section 3 by comparing the mean errors from the simulation values with the analytic estimates. In addition, we study the effects of C_a and C_s in isolation from each other. We chose the following three broad categories of experiments, i.e., models. In

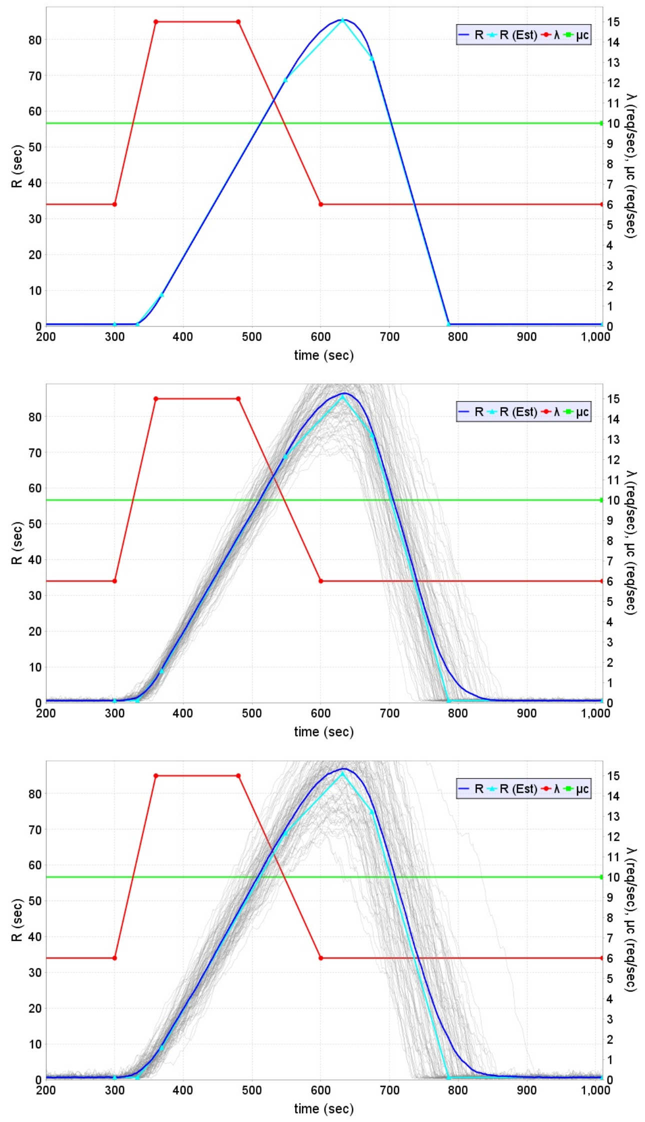

Figure 7: Comparison of estimates with simulation for trapezoidal surge ($\alpha \neq 0$ & $\gamma \neq 0$). Model: {top: $D/D/5$, middle: $M/M/5$, bottom: $G/G/5$; C_a= 1.4; C_s=1.3;}; μ= 2.0 req/sec; ρ_1= 0.60; ρ_2= 1.50; R_1= {top: 0.50, middle: 0.56, bottom: 0.61} sec; α= 60 sec; β= 180 sec; γ= 120 sec; averaged over 100 independent runs

addition, this study also meets the need for investigation of accuracy of estimate equations using a wide range of input parameters.

1. $G/D/c$: effects of C_a alone because C_s=0;
2. $D/G/c$: effects of C_s alone because C_a=0;
3. $G/G/c$: effects of C_a and C_s together.

We designed a set of experiments for each model by varying the values of C_a, C_s, ρ_2, and β while keeping $c = 5$, $\alpha = 45$ sec, and $\gamma = 75$ sec for all experiments.

- $C_a \in \{0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0\}$; or 0 for $D/G/5$;

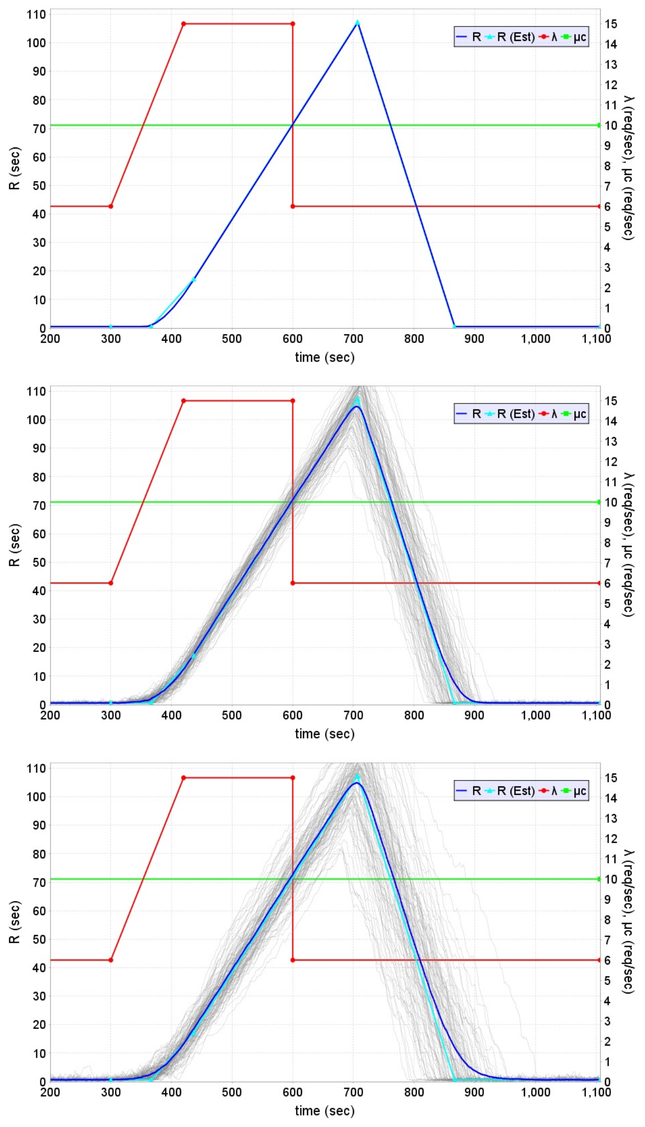

Figure 8: Comparison of estimates with simulation for trapezoidal surge ($\alpha \neq 0$ & $\gamma = 0$). Model: {top: $D/D/5$, middle: $M/M/5$, bottom: $G/G/5$; C_a= 1.4; C_s=1.3;}; μ= 2.0 req/sec; ρ_1= 0.60; ρ_2= 1.50; R_1= {top: 0.50, middle: 0.56, bottom: 0.61} sec; α= 120 sec; β= 180 sec; γ= 0 sec; averaged over 100 independent runs

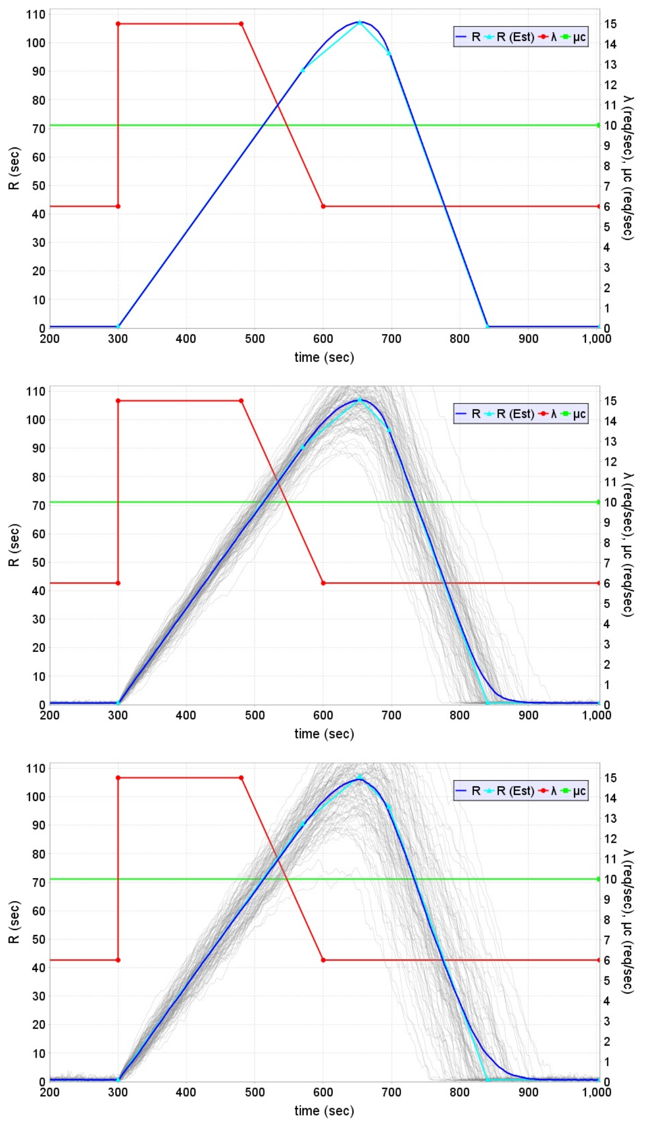

Figure 9: Comparison of estimates with simulation for trapezoidal surge ($\alpha = 0$ & $\gamma \neq 0$). Model: {top: $D/D/5$, middle: $M/M/5$, bottom: $G/G/5$; C_a= 1.4; C_s=1.3;}; μ= 2.0 req/sec; ρ_1= 0.60; ρ_2= 1.50; R_1= {top: 0.50, middle: 0.56, bottom: 0.61} sec; α= 0 sec; β= 180 sec; γ= 120 sec; averaged over 100 independent runs

- $C_s \in \{0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0\}$; or 0 for $G/D/5$;
- $C_a = C_s$ for all $G/G/5$ experiments;
- $\rho_1 = 0.6$;
- $\rho_2 \in \{1.25, 1.50, 1.75, 2.00\}$;
- $\beta \in \{60, 120, 180, 240, 300\}$ sec

All combinations of these input variables resulted in 140 experiments for each model and each experiment consisted of 100 runs each. Thus, we conducted a total of 420 experiments or 42,000 runs in total. We refer to each of these three sets of 140 experiments as a *set* of experiments in our discussion. The total number of requests processed during these simulations was around 186 million for each model.

Table 3 shows the mean and 95% confidence intervals for the errors of the metrics for the three *sets* of experiments.

We summarize our observations below based on the statistical results presented on Table 3.

- The mean absolute error ranged from 1.80% [for $G/G/5 : \epsilon(\delta)$] to 4.24% [for $G/G/5 : \epsilon(R_2)$]. Hence, the analytic estimates are good proxies for the actual results of a *set* of experiments. Because the widths

Table 3: Mean and 95% Confidence Intervals for a *set* of Experiments' Errors of Metric Estimate

	ϵ (metric)	$G/D/5$	$D/G/5$	$G/G/5$
1	$\epsilon(R_2)$	3.14% $\pm$ 0.5%	2.89% $\pm$ 0.5%	4.24% $\pm$ 0.6%
2	$\epsilon(\delta)$	2.24% $\pm$ 2.0%	2.60% $\pm$ 1.6%	1.80% $\pm$ 2.2%
3	$\epsilon(\Delta)$	3.39% $\pm$ 0.4%	3.16% $\pm$ 0.4%	4.00% $\pm$ 0.5%

of the 95% confidence intervals for the mean values of the errors are very small, this provides further validation to the formulas.

- Combinations of input parameters with high values of $\alpha+\beta+\gamma$, ρ_2 and low values of C_a and C_s resulted in lower errors.
- Higher but acceptable errors were obtained for combinations of input parameters with low values of $\alpha+\beta+\gamma$ and ρ_2 with higher values of C_a and C_s.
- The errors for D/D/5 are insignificant or very close to zero for many combinations of $\alpha+\beta+\gamma$, $\rho_1 < 1.0$ and $\rho_2 > 1.0$.
- No significant differences in the errors were observed due to either C_a or C_s or both.
- In general, the 95% confidence intervals ranged from low single digit % to mid single digit %.

Based on all the above observations one can conclude that the estimate formulas proposed for the three metrics are in close agreement with the experimental/simulation results for a wide range of values of ρ_2, β, C_a, and C_s.

5.3 Comparing Estimates with Simulation for Triangular Surges

Figures 10, 11, and 12 illustrate the pictorial comparison of the workload surge estimates with those of simulation experiments for three different sets of triangular shaped workload surges. Each set consists of $D/D/5$, $M/M/5$, and $G/G/5$ systems. In each of the figures, the cyan curve is the response time computed using the analytical estimates; the blue curve is the average response time computed using 100 independent simulation runs. The results clearly illustrate that the surge estimates are very close to those of simulation experiments.

6 Autonomic Elasticity Control

Elasticity control is a mechanism that dynamically changes the number of servers (aka horizontal scaling) or changes the capacity of the servers (aka vertical scaling) as needed. When servers are virtualized, it is relatively easy to change the number of virtual machines and/or their characteristics.

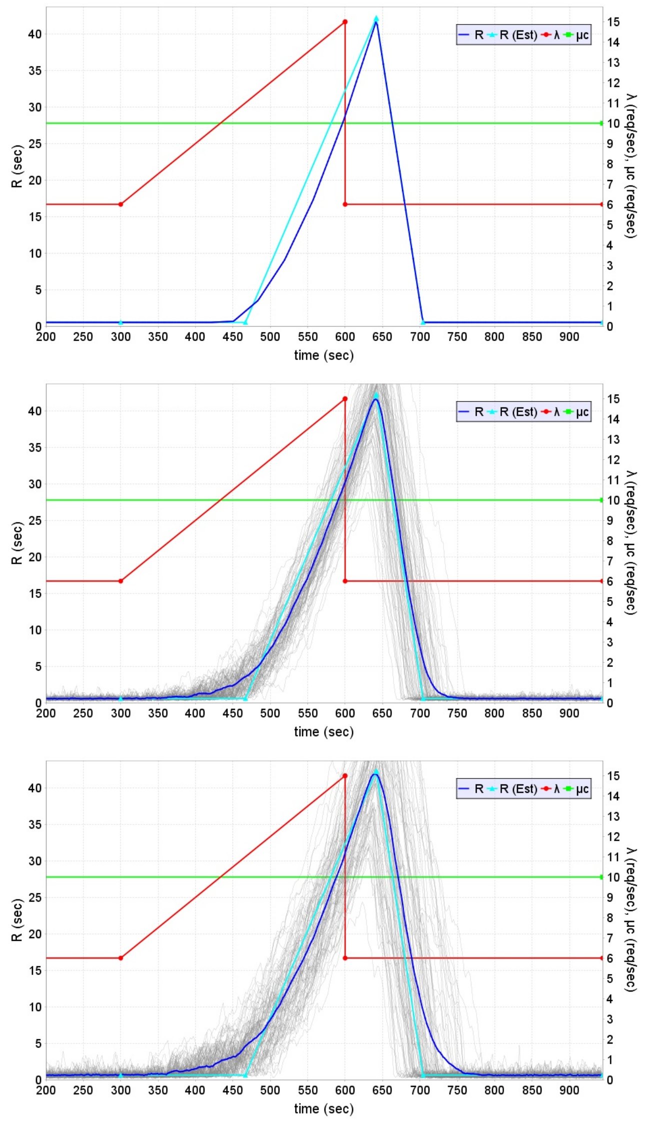

Figure 10: Comparison of estimates with simulation for triangular surge, Type 1 ($\alpha \neq 0$, $\beta = 0$ & $\gamma = 0$). Model: {top: $D/D/5$, middle: $M/M/5$, bottom: $G/G/5$; C_a= 1.4; C_s=1.3;}; μ= 2.0 req/sec; ρ_1= 0.60; ρ_2= 1.50; R_1= {top: 0.50, middle: 0.56, bottom: 0.61} sec; α= 300 sec; β= 0 sec; γ= 0 sec; averaged over 100 independent runs

We first consider an autonomic elasticity controller that performs horizontal scaling to maintain the peak response time below a certain threshold $R_{\max}$. Because one cannot predict the surge duration (α, β, and γ) ahead of time and because it is preferable not to overprovision, the controller monitors the surge duration and at regular intervals, uses the estimated peak response time equations (see Eqs. (23) and (30)) to estimate the minimum number of servers, $c_{\min}$, needed to maintain the peak response time below $R_{\max}$. Thus,

$$R_2 \approx (\rho_2 - 1)(\beta + \frac{K(\alpha+\gamma)}{2})$$

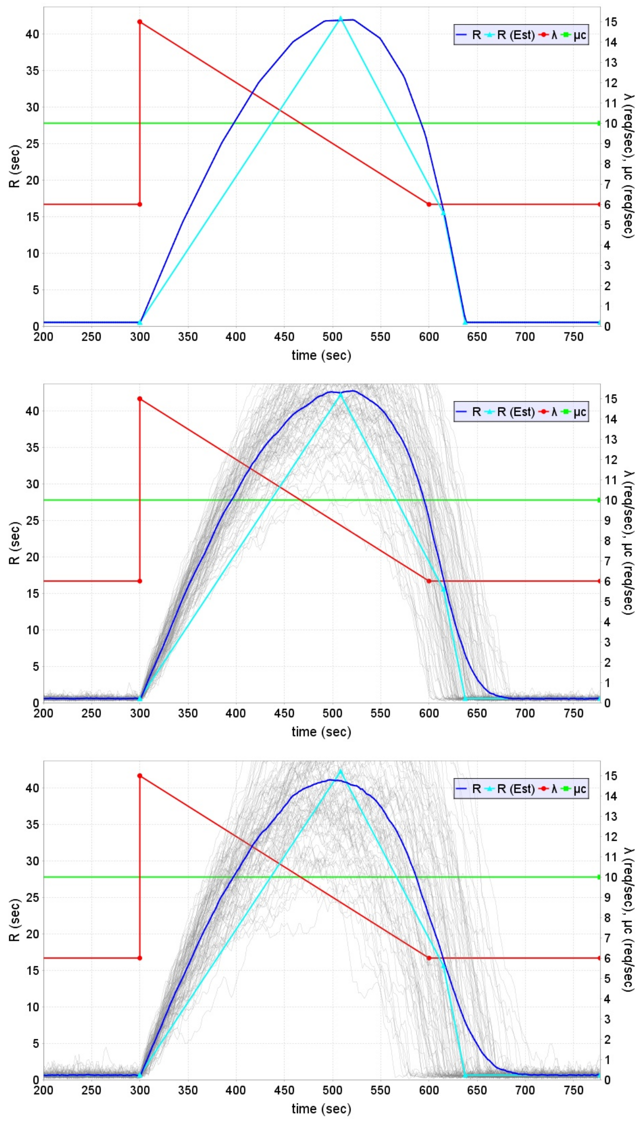

Figure 11: Comparison of estimates with simulation for triangular surge, Type 2 ($\alpha = 0,\ \beta = 0$ & $\gamma \neq 0$). Model: {top: $D/D/5$, middle: $M/M/5$, bottom: $G/G/5$; C_a= 1.4; C_s=1.3;}; μ= 2.0 req/sec; ρ_1= 0.60; ρ_2= 1.50; R_1= {top: 0.50, middle: 0.56, bottom: 0.61} sec; α= 0 sec; β= 0 sec; γ= 300 sec; averaged over 100 independent runs

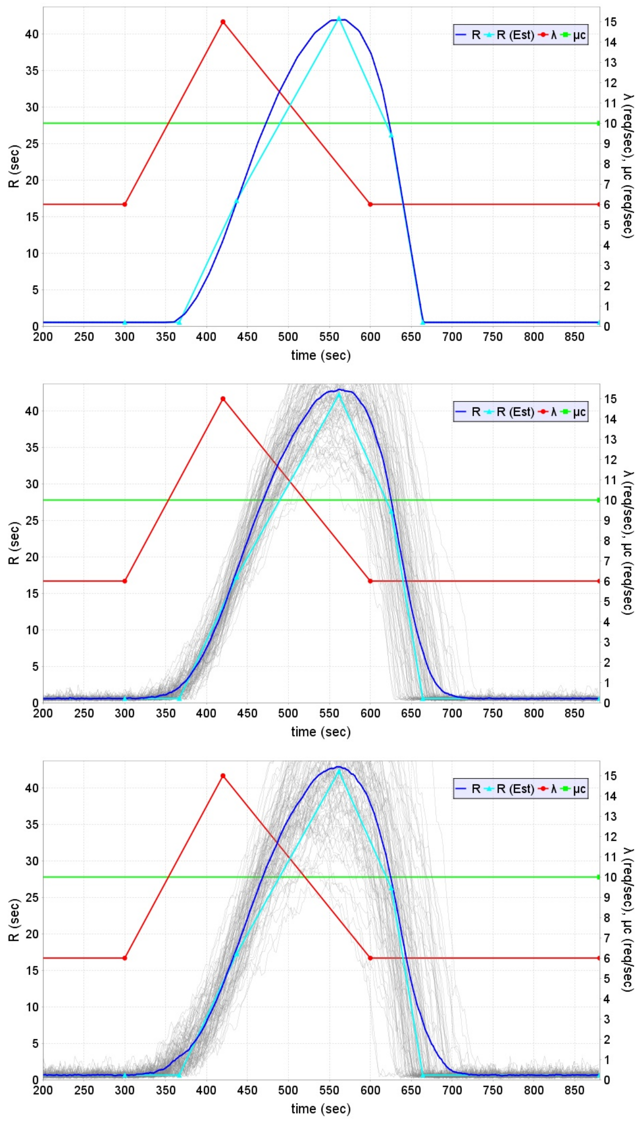

Figure 12: Comparison of estimates with simulation for triangular surge, Type 3 ($\alpha \neq 0,\ \beta = 0$ & $\gamma \neq 0$). Model: {top: $D/D/5$, middle: $M/M/5$, bottom: $G/G/5$; C_a= 1.4; C_s=1.3;}; μ= 2.0 req/sec; ρ_1= 0.60; ρ_2= 1.50; R_1= {top: 0.50, middle: 0.56, bottom: 0.61} sec; α= 120 sec; β= 0 sec; γ= 180 sec; averaged over 100 independent runs

$$\begin{aligned} &\leq R_{\max} \\ \Rightarrow \rho_2 &\leq \frac{R_{\max}}{\beta + K(\alpha+\gamma)/2} + 1 \\ \Rightarrow \frac{\lambda_2}{\mu c} &\leq \frac{R_{\max}}{\beta + K(\alpha+\gamma)/2} + 1 \\ \Rightarrow c &\geq \frac{\lambda/\mu}{\frac{R_{\max}}{\beta+K(\alpha+\gamma)/2} + 1} \end{aligned} \tag{42}$$

Because the number of servers has to be an integer, we get

$$c_{\min} = \left\lceil \frac{\lambda/\mu}{\frac{R_{\max}}{\beta+K(\alpha+\gamma)/2} + 1} \right\rceil . \tag{43}$$

For vertical scaling, we need to find the minimum value $\mu_{\min}$ of μ needed to maintain the peak response time below $R_{\max}$. Similarly to the equation above, we get,

$$\mu_{\min} = \frac{\lambda/c}{\frac{R_{\max}}{\beta+K(\alpha+\gamma)/2} + 1} . \tag{44}$$

If server capacities can only be selected from a discrete set such as in Amazon's EC2, the value of $\mu_{\min}$ can be used the determine the lowest capacity VM that is higher than $\mu_{\min}$.

Algorithm 1 shows the elasticity controller algorithm for horizontal scaling. The while loop between lines 2 and 21 includes the operation of the elasticity controller while the system is operational. The inputs used by the controller are: (1) the average arrival rate of requests λ assumed to be constantly monitored and available to the controller, (2) the service rate μ of each server, (3) the number c_{original} of original servers, and (4) the maximum desirable response time $R_{\max}$. The controller wakes up at regular intervals (called controller intervals) of duration τ, checks if the traffic intensity is below a surge level ($\rho < 1$; see lines 5-8) or experiencing a surge ($\rho \geq \rho_{\text{original}}$; lines 10-19). Algorithm 1 assumes that λ is the most recent average arrival rate of requests accumulated during the most recent controller interval. The variable CurrentTime has the current clock time and the function Sleep (t) suspends the operation of the controller for a time t. When the controller detects in line 5 that a surge has started, it moves to line 9 and records the time at which the surge started as the current time minus half of the sleep time τ. The reason for this adjustment is that the traffic intensity ρ could have become > 1 anytime while the controller was sleeping in line 6. On average, we assume the surge started midway during that time.

The controller assumes that a function called WorkloadAnalysis (line 12) continuously analyzes the workload intensity using any of a number of techniques (e.g., pattern recognition, time-series data mining, machine learning) and makes available the values of (α, β, γ) to the controller. Recognizing the features of the workload is orthogonal to elasticity control and is outside the scope of this paper. Because the controller sleeps for τ seconds within the loop, (α, β, γ) are incremented by τ every time. In line 13, the minimum number of servers $c_{\min}$ is recomputed with the updated values of (α, β, γ). As (α, β, γ) increase, $c_{\min}$ increases. After the surge, the number of servers is returned to the original value c_{original} (line 20). Note that if no surge occurs, the controller keeps executing the while loop in lines 5-8.

An algorithm for a vertical elasticity controller is given in Algorithm 2. This algorithm is very similar to the horizontal elasticity controller of Algorithm 1 with the server capacity μ used as a control knob instead of the number of servers.

Algorithm 1: Horizontal Elasticity Controller for Trapezoidal Shaped Workload Surges

```
   Input  : $\lambda, \mu, c_{\text{original}}, R_{\max}$
1  $\rho_{\text{original}} \leftarrow \lambda / (\mu \; c_{\text{original}})$
2  while system is online do
3      $c \leftarrow c_{\text{original}}$
4      $\rho \leftarrow \lambda / (\mu \; c)$
       /* wake up at regular intervals while
          surge has not started                    */
5      while $\rho < 1$ do
6          Sleep ($\tau$)
7          $\rho \leftarrow \lambda / (\mu c)$
8      end
9      SurgeStart $\leftarrow$ CurrentTime - $\tau/2$
       /* While surge is ongoing                   */
10     while $\rho \geq \rho_{\text{original}}$ do
11         $\rho \leftarrow \lambda / (\mu \; c)$
           /* Update surge durations               */
           /* ext. function returns $\alpha, \beta, \gamma$    */
12         $\alpha, \beta, \gamma \leftarrow$ WorkloadAnalysis ()
           /* Compute min. # of servers            */
13         $c_{\min} = \left\lceil \frac{\lambda/\mu}{(R_{\max}/(\beta + K(\alpha+\gamma)/2))+1} \right\rceil$
14         if $c_{\min} > c$ then
               /* Change # required servers        */
15             Change # servers in the system to $c_{\min}$
16             $c \leftarrow c_{\min}$
17         end
18         Sleep ($\tau$)
19     end
       /* return to original # servers             */
20     Change # servers in the system to $c_{\text{original}}$
21 end
```

7 Evaluation of the Controller

This section presents an experimental evaluation of the horizontal elasticity controller discussed in the previous section. The metrics used to assess the effectiveness of the controller are:

- D: time during which the response time suffered;

$$D = t'_{norm} - t_a = K\alpha + \beta + \gamma + \delta + \Delta.$$

- Φ: area under the curve of $R(t)$ during D; more specifically, $\Phi = \int_{t_a}^{t_a+D} R(t)dt$.

These metrics are identified in Fig. 13, which shows the behavior of a multi-server system during a workload surge, with and without the controller. We use the subscripts c and nc hereafter to indicate metrics obtained when the controller is on and off, respectively.

We also define a function, $\eta \; (X)$ of metric X to represent the percentage improvement of X when using the autonomic controller:

$$\eta \; (X) = 100 \times \frac{X_{nc} - X_c}{X_c}. \tag{45}$$

The three metrics of interest in our case are R_2, D, and Φ. Because $R_{2_{nc}} \geq R_{2_c}$, $D_{nc} \geq D_c$, and $\Phi_{nc} \geq \Phi_c$, the percentage improvement of these metrics is always positive.

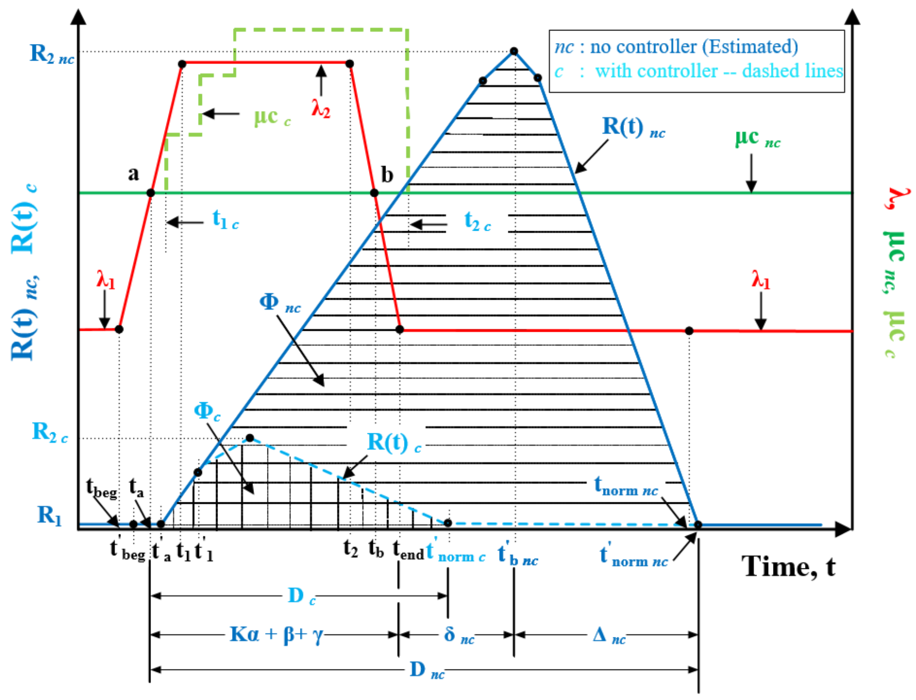

Figure 13: Comparison of the surge behavior with and without a controller

We give below expressions for the three metrics, for the case in which the controller is disabled, using the equations derived in the prior sections and using Fig. 13:

$$R_{2_{nc}} = (\rho_2 - 1)(\beta + \frac{K(\alpha+\gamma)}{2}) \quad (46)$$

$$D_{nc} = (K\alpha + \beta + \gamma) + \delta_{nc} + \Delta_{nc} \quad (47)$$

$$\Phi_{nc} = \int_{t_a}^{t_a + D_{nc}} R(t)\, dt \quad (48)$$

Φ_{nc} is obtained by computing the area under the curve by using Figs. 4 and 13 and the points (x_i, y_i) in Table 1 as:

$$\Phi_{nc} = \sum_{i=2}^{6} \frac{1}{2}(x_{i+1} - x_i) \times (y_{i+1} + y_i). \quad (49)$$

We conducted experiments with the horizontal elasticity controller and measured the three metrics described above and computed the improvement η as reported in Table 4. The no controller values in the table are obtained from Eqs. (46)-(48). The table reports three experiments for G/G/5 with increasing values of the surge duration (60 sec, 180 sec, and 300 sec). Then, it reports results for M/M/5 and D/D/5, both of them for surge durations $(\alpha + \beta + \gamma)$ of 420 sec. The first general observation is that the percent relative improvement $\eta()$ is very large (ranges from 298 to 47,274) for all three metrics and for all five scenarios. Also, the largest gains are for Φ, followed by R_2, and D. When we compare the three G/G/5 cases we see that the gains increase as the surge duration increases. This is a good property of the controller and it is a consequence of the fact that the controller incrementally adjusts the number of servers as it obtains a better estimate of the surge duration.

The behavior of the controller can be appreciated in Fig. 14 that illustrates (a) the surge (in red), (b) the predicted response time (in light blue), (c) the average over all runs of the response time with the controller enabled (in dark blue), (d) the individual values of the response times with the controller for all 100 runs (gray curves), and (e) the variation of the number of servers (in green). All figures are for a G/G/c case that starts with five servers and has coefficients of variation $C_a = 1.40$ and $C_s = 1.30$. The traffic intensity without the controller before the surge was $\rho_1 = 0.60$ and after the surge $\rho_2 = 1.50$. The five plots on the left correspond to $R_{\max} = 5$ sec and those on the right to $R_{\max} = 10$ sec.

In each side, $\alpha = 45$ sec, $\gamma = 75$ sec, and the value of β is 60 (top) and 300 (bottom) sec. The controller interval was set to $\tau = 15$ sec in all cases.

The following observations can be drawn from Fig. 14: (a) In both cases, the controller was able to substantially reduce the peak response time when compared to the response time peak without the controller. Higher reductions can be seen for higher values of the surge duration. For example, for $\alpha = 45$ sec, $\beta = 60$ sec, $\gamma = 75$ sec; and $R_{\max} = 10$ sec, the controller brought the response time peak from about 47 seconds to about 5 seconds. And for $\alpha = 45$ sec, $\beta = 300$ sec, $\gamma = 75$, the peak response time is reduced from 168 to 6 seconds with the controller. (b) The controller tries to minimize the cost of using more servers by incrementally adding more servers as it has a

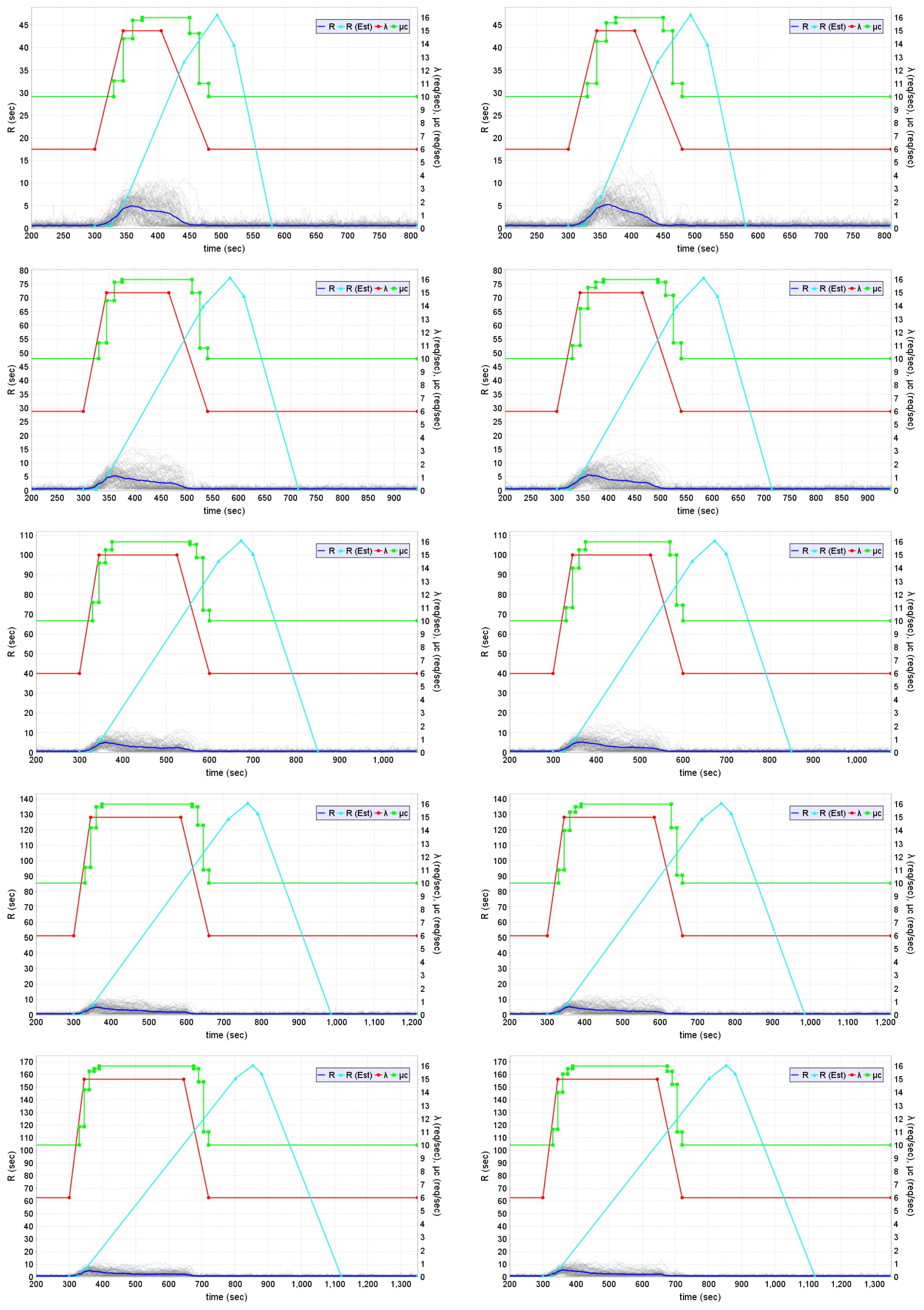

Figure 14: With Horizontal Elasticity Controller - $G/G/5; C_a = 1.40; C_s = 1.30; \rho_{1_{nc}} = 0.60; \rho_{2_{nc}} = 1.50; \alpha = 45$ sec; $\beta = \{60, 120, 180, 240, 300\}$ sec; $\gamma = 75$ sec; $\tau = 15$ sec; $R_{\max}$ ={left column= 5 sec; right column= 10 sec}; Average over 100 independent runs.

Algorithm 2: Vertical Elasticity Controller for Trapezoidal Shaped Workload Surges

```
Input   : $\lambda, \mu_{\text{original}}, c, R_{\max}$
$\rho_{\text{original}} \leftarrow \lambda/(\mu_{\text{original}}\, c)$
while system is online do
    $\mu \leftarrow \mu_{\text{original}}$
    $\rho \leftarrow \lambda/(\mu\, c)$
    /* wake up at regular intervals while
       surge has not started                 */
    while $\rho < 1$ do
        Sleep ($\tau$)
        $\rho \leftarrow \lambda/(\mu c)$
    end
    SurgeStart $\leftarrow$ CurrentTime - $\tau/2$
    /* While surge is ongoing                */
    while $\rho \geq \rho_{\text{original}}$ do
        $\rho \leftarrow \lambda/(\mu\, c)$
        /* Update surge durations            */
        /* ext. function returns $\alpha, \beta, \gamma$     */
        $\alpha, \beta, \gamma \leftarrow$ WorkloadAnalysis ()
        /* Compute min. server capacity      */
        $\mu_{\min} = \frac{\lambda/c}{(R_{\max}/(\beta+K(\alpha+\gamma)/2))+1}$
        if $\mu_{\min} > \mu$ then
            /* Update server capacity        */
            Change server capacity to $\mu_{\min}$
            $\mu \leftarrow \mu_{\min}$
        end
        Sleep ($\tau$)
    end
    /* return to original capacity           */
    Change server capacity to $\mu_{\text{original}}$
end
```

better estimate of the surge duration.

As Fig. 14 indicates, the number of servers is continually increased by the controller in order to cope with the surge. It is thus important to quantify the cost incurred by these additional servers. We define the metric κ as the number of additional server-seconds used by the controller (see Fig. 13):

$$\kappa = \int_{t_{1_c}}^{t_{2_c}} [c_c(t) - c_{nc}] \; dt \tag{50}$$

where t_{1_c} and t_{2_c} delimit the time during which the controller added more resources, $c_c(t)$ is the number of resources used by the controller at time t, and c_{nc} is the fixed number of resources used when the controller is disabled. For illustration purposes, Table 5 shows the value of κ for a G/G/5 system with $C_a = 1.4; C_s = 1.3$; $\lambda_1 = 6$ req/sec; $\lambda_2 = 15$ req/sec; and $\mu = 2$ req/sec, several surge durations ($\alpha = 45$ sec, $\gamma = 75$ sec, $\beta \in \{60, 120, 180, 240, 300\}$) and two values of $R_{\max}$ for two different values of the controller interval time τ. The table indicates that (1) The additional server-seconds κ increases with the surge duration because there is a need to keep more resources for a longer time interval. (2) A more stringent SLA ($R_{\max} = 5$ vs. $R_{\max} = 10$) requires more resources during longer time. (3) A more responsive controller ($\tau = 15$ vs. $\tau = 30$) provides more resources to counter the surge.

Table 4: Performance of the Horizontal Elasticity Controller - Improvements achieved; η's of Metrics

[a] $G/G/c; C_a = 1.4; C_s = 1.3; \lambda_1 = 6$ req/sec; $\lambda_2 = 15$ req/sec; $\mu = 2$ req/sec; $c_{nc} = 5; c_c =$ varies; $\tau = 15$ sec; $R_{max} = 10$ sec; $\alpha = 45$ sec; $\beta = 60$ sec; $\gamma = 75$ sec				
		No controller	With Controller	$\eta()\%$
1	R_2 (sec)	46.7	7.2	549
2	D (sec)	260.0	65.4	298
3	Φ (sec^2)	6,553.0	329.0	1,892
[b] $G/G/c$ with the same parameters as above and $\beta = 180$ sec				
		No controller	With Controller	$\eta()\%$
1	R_2 (sec)	106.7	6.7	1,493
2	D (sec)	530.0	91.8	477
3	Φ (sec^2)	29,530.0	334.0	8,741
[c] $G/G/c$ with the same parameters as above and $\beta = 300$ sec				
		No controller	With Controller	$\eta()\%$
1	R_2 (sec)	166.7	6.9	2,316
2	D (sec)	800.0	98.7	711
3	Φ (sec^2)	68,707.0	351.0	19,475
[d] $M/M/c; \lambda_1 = 6$ req/sec; $\lambda_2 = 15$ req/sec; $\mu = 2$ req/sec; $c_{nc} = 5; c_c =$ varies; $\tau = 15$ sec; $R_{max} = 10$ sec; $\alpha = 45$ sec; $\beta = 300$ sec; $\gamma = 75$ sec				
		No controller	With Controller	$\eta()\%$
1	R_2 (sec)	166.7	6.1	2,633
2	D (sec)	800.0	81.6	880
3	Φ (sec^2)	68,701.0	312.0	21,920
[e] $D/D/c; \lambda_1 = 6$ req/sec; $\lambda_2 = 15$ req/sec; $\mu = 2$ req/sec; $c_{nc} = 5; c_c =$ varies; $\tau = 15$ sec; $R_{max} = 10$ sec; $\alpha = 45$ sec; $\beta = 300$ sec; $\gamma = 75$ sec				
		No controller	With Controller	$\eta()\%$
1	R_2 (sec)	166.7	3.5	4,663
2	D (sec)	800.0	64.0	1,150
3	Φ (sec^2)	68,693.0	145.0	47,274

Table 5: Additional resources utilized by the horizontal elasticity controller, κ (server.sec)

$G/G/c; C_a = 1.4; C_s = 1.3; \lambda_1 = 6$ req/sec; $\lambda_2 = 15$ req/sec; $\mu = 2$ req/sec; $c_{nc} = 5; c_c =$ varies; α= 45 sec; γ= 75 sec				
	$\tau = 15$ sec		$\tau = 30$ sec	
β (sec)	$R_{\max} = 5$	$R_{max} = 10$	$R_{\max} = 5$	$R_{\max} = 10$
60	377	369	348	342
120	620	618	558	531
180	858	816	777	735
240	1,148	1,113	990	957
300	1,404	1,335	1,209	1,170

We also conducted experiments to demonstrate the robustness of the controller even when the workload does not have an exact trapezoidal shape as assumed for the derivation of the equations used by the controller. For that purpose, we injected random upward and down-

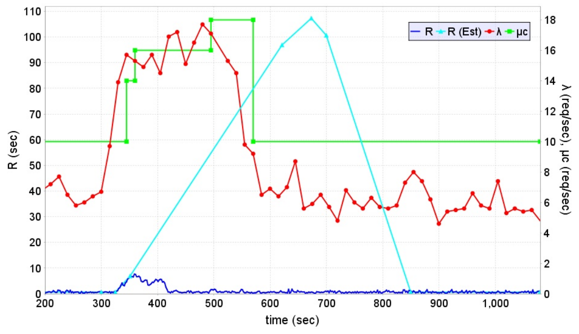

Figure 15: Controller effect under a 5% perturbation of the average arrival rate.

ward perturbations every τ seconds to the average workload intensity outside and during the surge. These modified average workload intensity values drive the generation of the interarrival times during each interval of τ seconds. For example, Fig. 15 shows that the controller correctly detects the onset of the surge, increases the number of servers three times during the surge, and decreases the number of servers to its original value when it detects the end of the surge.

8 Related Work

The authors of [15] conducted a comprehensive study on elasticity mechanisms by proposing a classification based on scope, policy, purpose and method. Under this classification, our work falls: (1) Scope /Infrastructure, (2) Purpose/Performance, (3) Policy/Automatic/Predictive, and (4) Method/Redimensioning. The work in [16] describes a broker to acquire resources on demand from a public cloud to service requests from a client enterprise. Reactive auto-scaling is performed by the broker based on user demand for the IaaS resources. The goal of the broker is to maximize the profit for the intermediary enterprise while attempting to reduce the cost for the client enterprise. That work does not present any equations that can be used to predict the effects of workload intensity surges. The work in [17] presented an elasticity management framework that takes the input typically presented to reactive rule-based scaling strategies and returns a proactive auto scaler. Their elasticity management framework combines reactive and proactive techniques and uses data mining for prediction purposes. Examples of predictive control can be found in [18]. The authors of [19] evaluated various auto-scaling strategies using log traces from a Google's data center cluster comprising of millions of jobs using the utilization level as a key performance indicator. They show that proper management of the parameters of an auto-scaling strategy reduces the difference between the target utilization and the actual values. There are many challenges in cloud autoscaling including the need to accurately estimate resource usage in the face of significant variability in workload patterns as discussed in [20], which presents a model-predictive algorithm for workload forecasting for resource autoscaling. The sensitivity of auto-scaling mechanisms to prediction results has been investigated in [21]. Their work compared threshold-based scaling techniques based on Support Vector Machine (SVM) and Neural Networks (NN) predictions. The work in [22] provided an extensive review of reactive and proactive autoscaling techniques and discussed in detail five different groups: static, threshold-based policies, reinforcement learning, queuing theory, control theory, and time-series analysis. The authors of [23] presented a color set algorithm for autoscaling of Internet applications for cloud computing that achieved good demand satisfaction ratio and saved energy by reducing the number servers used when the load is low. The advantages of elasticity depend to some extent on the delays involved in resource provisioning. A study on the startup time of cloud VMs across three cloud providers, Amazon EC2, Windows Azure and Rackspace, was described in [24]. The authors of [25] presented CloudPerf, a performance test framework designed for distributed and dynamic multi-tenant environments. CloudPerf has features for elasticity in cloud environments. The authors of [26] designed and evaluated a proactive and application-aware auto-scaler using an ensemble of open-source tools available online. They used those tools for forecasting of arrival rates, resource demand estimation, and software performance modeling of the application. Autoscaling techniques have been applied to many different types of applications. The work in [27] investigated elastic scaling for stream processing applications deployed in private clouds. They have developed an elastic switching mechanism to reduce the latency of event processing jobs by scaling up using resources from a public cloud. The authors in [28] have shown that a hybrid controller using horizontal elasticity which incorporated a reactive controller for scale up coupled with proactive controllers for scale down decisions reduced SLA violations significantly compared with just reactive controllers. The authors in [29] conducted experimental studies to compare the performance of autoscaling policies applied to applications modeled as workflows, i.e., applications modeled as directed acyclic graphs. They evaluated seven different policies on three scientific applications and highlighted the trade-offs between these policies. The authors in [28] showed that a hybrid controller using horizontal elasticity that incorporates a reactive controller for scaling up coupled with proactive controllers for scale down decisions reduced SLA violations significantly compared with just reactive controllers. The authors of [30] proposed EStream, an elastic batched stream processing system, which adjusts available resources to handle workload fluctuation in container cloud. The authors of [31] implemented X-Graph, a distributed graph computing prototype in the cloud. Their system demonstrated the elastic capabilities in multiple ways. The authors of [32] modeled the cloud infrastructure as $G/G/c$ queue in steady state and then proposed an autonomic elasticity controller that changed the number of virtual machines allocated to a service in the cloud based on monitored load changes and predictions of future load. The authors of [33] proposed a hybrid elasticity approach that takes advantage of both a capacity based approach and performance based approach. Extensive research has been done to improve the performance of applications in the cloud by utilizing elasticity or new designs for existing services. For example, the work in [34] proposed two new designs to improve the performance and scalability of OpenStack Swift, an object storage service, which is widely used for cloud-based distributed storage.

9 Concluding Remarks and Future Work

Many computer systems, such as Internet datacenters and cloud computing environments, consist of a multitude of servers that process user requests and may be subject to surges in the traffic intensity during intervals of time when the offered load exceeds the system's capacity. We derived a generic property for surges of any shape and from there obtained, for trapezoidal and triangular surges, equations to estimate the impact of a surge on the peak response time and on the time lag between the end of the surge and the time at which the response time peaks. These equations were extensively validated using simulation, which showed very small errors between the analytic estimates and simulation.

We then designed and implemented elastic controllers (a horizontal and a vertical) that use the derived expressions to estimate the number or capacity of required resources to meet response time SLAs while keeping the increase in server capacity to the minimum necessary. Several experiments with the controller indicate that it meets its goals. The duration of the controller interval τ influences the maximum response time R_{2c}.

We are currently working on several extensions to the work reported here. First, we are incorporating server startup delays into our controller since most resources (e.g., VMs in the cloud) cannot be provisioned instantaneously; we are in the process of designing strategies to counter the negative effects of server startup delays. Second, we are analyzing the Google traces [35] to characterize patterns of workload surges that we may use to further evaluate elasticity controllers. The study of these patterns might shed some light on the shapes of these surges. We intend to analyze surge patterns that are more general than the trapezoidal ones described here, which include rectangular and triangular. However, we showed in Fig. 15 that our controller works well even when the workload surge is not exactly trapezoidal. We feel that our work is an important first step towards generalizing model-driven controllers given that many practical workload surges may be synthesized from a combinations of trapezoidal shaped surges and other simple patterns.

Acknowledgements

The work of Daniel A. Menascé was partially supported by the AFOSR grant FA9550-16-1-0030.

References

[1] L. Kleinrock, *Queueing Systems. Vol.1: Theory*. Wiley-Interscience, 1975.

[2] D. A. Menascé, L. W. Dowdy, and V. A. F. Almeida, *Performance by Design:computer capacity planning by example*. Prentice Hall, 2004.

[3] V. Tadakamalla and D. Menascé, "Analysis and autonomic elasticity control for multi-server queues under traffic surges," in *Cloud and Autonomic Computing (ICCAC), 2017 Intl. Conf.* IEEE, 2017, pp. 92–103.

[4] J. Jung, B. Krishnamurthy, and M. Rabinovich, "Flash crowds and denial of service attacks: Characterization and implications for CDNs and web sites," in *Proc. 11th Intl. Conf. World Wide Web*, ser. WWW '02. New York, NY, USA: ACM, 2002, pp. 293–304.

[5] I. Ari, B. Hong, E. L. Miller, S. A. Brandt, and D. D. E. Long, "Managing flash crowds on the internet," in *11th IEEE/ACM Intl. Symp. Modeling, Analysis and Simulation of Computer Telecommunications Systems, 2003. MASCOTS 2003.*, Oct 2003, pp. 246–249.

[6] D. Menascé, V. Almeida, R. Riedi, F. Ribeiro, R. Fonseca, and W. M. Jr., "A hierarchical and multiscale approach to analyze e-business workloads," *Performance Evaluation*, vol. 54, no. 1, pp. 33 – 57, 2003.

[7] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A berkeley view of cloud computing," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, Feb 2009.

[8] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *J. Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, 2010.

[9] Y. Ogawa, G. Hasegawa, and M. Murata, "Cloud bursting approach based on predicting requests for business-critical web systems," in *2017 Intl. Conf. Computing, Networking and Communications (ICNC)*, Jan 2017, pp. 437–441.

[10] Y. Kwok, P. Teller, and S. Arunagiri, "2tl: a scheduling algorithm for meeting the latency requirements of bursty i/o streams at user-specified percentiles," in *2017 IEEE Intl. Conf. Cloud and Autonomic Computing*, ser. ICCAC 2017, 2017.

[11] D. Gross, J. Shortle, J. Thompson, and C. Harris, "Fundamentals of queuing theory, 4th edition," *John Wiley & Sons*, 2008.

[12] N. R. Herbst, S. Kounev, and R. Reussner, "Elasticity in cloud computing: What it is, and what it is not," in *Proc. 10th Intl. Conf. Autonomic Computing (ICAC 13)*. San Jose, CA: USENIX, 2013, pp. 23–27.

[13] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *IEEE Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[14] M. Harchol-Balter, *Performance modeling and design of computer systems: queueing theory in action*. Cambridge University Press, 2013.

[15] G. Galante and L. C. E. d. Bona, "A survey on cloud computing elasticity," in *2012 IEEE Fifth Intl. Conf. Utility and Cloud Computing*, Nov 2012, pp. 263–270.

[16] A. Biswas, S. Majumdar, B. Nandy, and A. El-Haraki, "An auto-scaling framework for controlling enterprise resources on clouds," in *15th IEEE/ACM Intl. Symp. Cluster, Cloud and Grid Computing*, May 2015, pp. 971–980.

[17] L. R. Moore, K. Bean, and T. Ellahi, "Transforming reactive auto-scaling into proactive auto-scaling," in *Proc. 3rd Intl. Workshop on Cloud Data and Platforms*. New York, NY, USA: ACM, 2013, pp. 7–12.

[18] G. Mencagli, "Adaptive model predictive control of autonomic distributed parallel computations with variable horizons and switching costs," *Concurrency and Computation: Practice and Experience*, vol. 28, no. 7, pp. 2187–2212, 2016, cpe.3495.

[19] M. A. S. Netto, C. Cardonha, R. L. F. Cunha, and M. D. Assuncao, "Evaluating auto-scaling strategies for cloud computing environments," in *2014 IEEE 22nd Intl. Symp. Modelling, Analysis Simulation of Computer and Telecommunication Systems*, Sept 2014, pp. 187–196.

[20] N. Roy, A. Dubey, and A. Gokhale, "Efficient autoscaling in the cloud using predictive models for workload forecasting," in *2011 IEEE 4th Intl. Conf. Cloud Computing*, July 2011, pp. 500–507.

[21] A. Y. Nikravesh, S. A. Ajila, and C. H. Lung, "Measuring prediction sensitivity of a cloud auto-scaling system," in *2014 IEEE 38th Intl. Computer Software and Applications Conf.*, July 2014, pp. 690–695.

[22] T. Lorido-Botran, J. Miguel-Alonso, and J. A. Lozano, "A review of auto-scaling techniques for elastic applications in cloud environments," *Journal of Grid Computing*, vol. 12, no. 4, pp. 559–592, Dec 2014.

[23] Z. Xiao, Q. Chen, and H. Luo, "Automatic scaling of internet applications for cloud computing services," *IEEE Tr. Computers*, vol. 63, no. 5, pp. 1111–1123, May 2014.

[24] M. Mao and M. Humphrey, "A performance study on the vm startup time in the cloud," in *2012 IEEE 5th Intl.Conf. Cloud Computing*, June 2012, pp. 423–430.

[25] N. Michael, N. Ramannavar, Y. Shen, S. Patil, and J.-L. Sung, "Cloudperf: A performance test framework for distributed and dynamic multi-tenant environments," in *Proc. 8th ACM/SPEC Intl. Conf. Performance Engineering*, ser. ICPE '17. New York, NY, USA: ACM, 2017, pp. 189–200. [Online]. Available: http://doi.acm.org/10.1145/3030207.3044530

[26] A. Bauer, N. Herbst, and S. Kounev, "Design and evaluation of a proactive, application-aware auto-scaler: Tutorial paper," in *Proc. 8th ACM/SPEC on Intl. Conf. Performance Engineering*, ser. ICPE '17. New York, NY, USA: ACM, 2017, pp. 425–428.

[27] S. Ravindra, M. Dayarathna, and S. Jayasena, "Latency aware elastic switching-based stream processing over compressed data streams," in *Proc. 8th ACM/SPEC Intl. Conf. Performance Engineering*, ser. ICPE '17. New York, NY, USA: ACM, 2017, pp. 91–102.

[28] A. Ali-Eldin, J. Tordsson, and E. Elmroth, "An adaptive hybrid elasticity controller for cloud infrastructures," in *2012 IEEE Network Operations and Management Symp.*, April 2012, pp. 204–212.

[29] A. Ilyushkin, A. Ali-Eldin, N. Herbst, A. V. Papadopoulos, B. Ghit, D. Epema, and A. Iosup, "An experimental performance evaluation of autoscaling policies for complex workflows," in *Proc. 8th ACM/SPEC on Intl. Conf. Performance Engineering*, ser. ICPE '17. New York, NY, USA: ACM, 2017, pp. 75–86.

[30] S. Wu, X. Wang, H. Jin, and H. Chen, *Elastic Resource Provisioning for Batched Stream Processing System in Container Cloud*. Cham: Springer International Publishing, 2017, pp. 411–426.

[31] L. Lu, X. Shi, and H. Jin, *Towards Truly Elastic Distributed Graph Computing in the Cloud*. Cham: Springer International Publishing, 2015, pp. 300–309. [Online]. Available: https://doi.org/10.1007/978-3-319-26979-5_23

[32] A. Ali-Eldin, M. Kihl, J. Tordsson, and E. Elmroth, "Efficient provisioning of bursty scientific workloads on the cloud using adaptive elasticity control," in *Proceedings of the 3rd Workshop on Scientific Cloud Computing*, ser. ScienceCloud '12. New York, NY, USA: ACM, 2012, pp. 31–40. [Online]. Available: http://doi.acm.org/10.1145/2287036.2287044

[33] S. Farokhi, P. Jamshidi, E. Bayuh Lakew, I. Brandic, and E. Elmroth, "A hybrid cloud controller for vertical memory elasticity," *Future Gener. Comput. Syst.*, vol. 65, no. C, pp. 57–72, Dec. 2016.

[34] S. Gugnani, X. Lu, and D. K. D. Panda, "Swift-x: Accelerating openstack swift with rdma for building an efficient hpc cloud," in *Proceedings of the 17th IEEE/ACM Intl. Symp. Cluster, Cloud and Grid Computing*, ser. CCGrid '17. Piscataway, NJ, USA: IEEE Press, 2017, pp. 238–247.

[35] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: format+ schema," *Google Inc., Technical Report*, pp. 1–14, Nov 2011.