Analysis and Autonomic Elasticity Control for Multi-Server Queues Under Traffic Surges

Venkat Tadakamalla
Computer Science Department
George Mason University
Fairfax, VA, USA
vtadakam@masonlive.gmu.edu

Daniel A. Menascé
Computer Science Department
George Mason University
Fairfax, VA, USA
menasce@gmu.edu

***Abstract*—Many computing environments consist of a multitude of servers that process requests that arrive from a population of customers. 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. In this paper we study multi-server systems that are subject to *workload surges* during which time the average arrival rate of requests exceeds the system's capacity. This paper's main contributions are (1) The derivation of a set of equations to estimate the impact of workload surges on response time; (2) A simulator for a G/G/c system to evaluate the accuracy of the equations in (1); and (3) The design, implementation, and extensive evaluation of an autonomic controller for multi-server elasticity that uses the equations derived in (1). The results show that our equations estimate with great accuracy the impact of surges on response time and that our autonomic controller is able to successfully determine how to vary the number of servers to mitigate the impact of workload surges.**

***Index Terms*—elasticity control; cloud computing; autonomic computing; queuing theory; G/G/c; workload surge;**

I. Introduction

Many computing environments consist of a multitude of servers that process requests that arrive from a population of customers (e.g., web sites with multiple web servers at the front tier). Each request is served by one server only. When all servers are busy serving requests, arriving requests have to wait in a waiting line until a server becomes available. 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 for 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).

There is an extensive literature on the study of analytical models of queueing systems 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]–[3]). 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 results apply only to systems in steady state.

However, most actual systems are subject to *workload surges* (aka flash crowds), i.e, periods during which the arrival rate exceeds the system's capacity (see e.g., [4]–[9]). When that happens, the queue length grows continuously and so does the response time of requests. It turns out that the response time continues to increase even after the surge is finished. In other words, the response time does not return to its steady state value as soon as the surge is over. As an illustration, consider Fig. 1 that shows a rectangular-shaped 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 traffic intensity shows a surge from a value of 5 to 20 requests/sec and lasts for 5 minutes. 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 290 seconds was observed at time 880 sec and it only returned to its pre-surge level at time 1,260 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 [10]. 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 (see e.g., Amazon Elastic Compute Cloud, EC2). Elasticity has been defined as *the degree to which a system is able to adapt to workload changes by provisioning*

978-1-5386-1939-1/17 $31.00 © 2017 IEEE
DOI 10.1109/ICCAC.2017.16

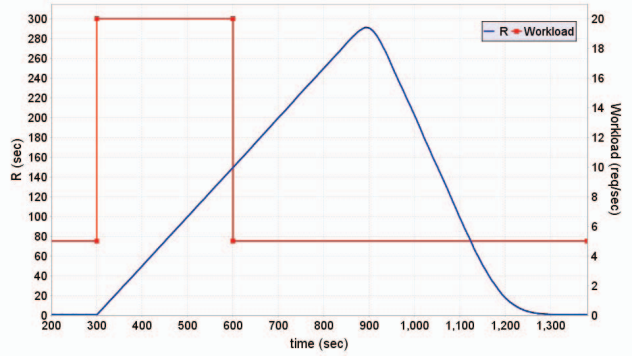

Fig. 1. Example of workload surge and corresponding effect on the response time for 5 servers. Workload surge duration: 5 minutes; average service time: 0.5 sec; average arrival rate before and after surge: 5.0 requests/sec; average arrival rate during surge: 20.0 requests/sec; coefficient of variation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

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 [11].

Most of the 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; while additional resources have not been deployed, degraded performance will be experienced by users. (2) because it is not straightforward to determine how many resources have to be added or released as the traffic intensity varies, there is a risk of overprovisioning or underprovisioning.

Therefore, there is a need to control the system's capacity in an autonomic manner [12] so that the system 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 set of equations to estimate the impact of workload surges on response time; (2) A simulator for a G/G/c system to evaluate the accuracy of the equations mentioned in (1); and (3) The design, implementation, and extensive evaluation, using the G/G/c simulator, of an autonomic controller for server elasticity .

The rest of this paper is organized as follows. Section II presents the basic concepts and notation used in the paper. Section III presents a set of analytical expressions that can be used to estimate the effects of a workload surge. These analytic expressions are thoroughly validated in Section IV through simulation. Section V 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 VI. Section VII discusses related work. Finally, Section VIII concludes the paper and discusses future work.

II. Basic Concepts and Notation

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
- 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 of the standard deviation of the interrarival time by its average.
- C_s: coefficient of variation of the service time, i.e., the ratio of the standard deviation of the service time by its average.
- ρ: traffic intensity. $\rho = \lambda/(\mu c)$.

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) [13]. Our simulation experiments show results for different values of C_a and C_s.

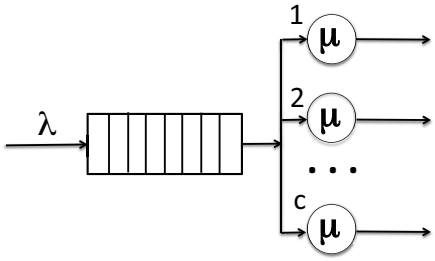

Fig. 2. Diagram of a G/G/c queue

III. Estimating the Effect of the Surge

In this section we derive analytic expressions for the following metrics and use Fig. 3 to illustrate the notation:

- $R_2 = R(t_2')$: estimate for the peak response time.
- $\delta = t_2' - t_2$: estimate for the peak response time lag, i.e., the time after the surge finished until the response time achieves its peak.

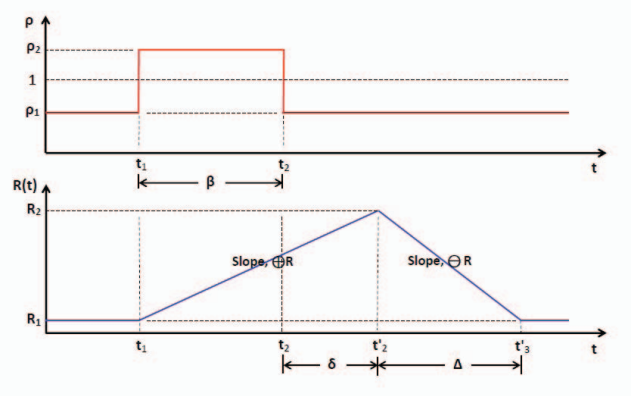

Fig. 3. Rectangular workload surge. Top: variation of the traffic intensity. Bottom: variation of the response time

- $\Delta = t_3^{'} - t_2^{'}$: 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.
- $\oplus R$: estimated increase rate of $R(t)$ during $t_1 \leq t \leq t_2^{'}$.
- $\ominus R$: estimated decrease rate of $R(t)$ during $t_2^{'} \leq t \leq t_3^{'}$.

We consider the rectangular-shaped workload surge shown in Fig. 3. Let the average arrival rate before the workload surge be λ_1; therefore the traffic intensity is $\rho_1 = \lambda_1/(\mu c)$. During the surge, the average arrival rate is considered to be λ_2 and therefore the traffic intensity during the surge is $\rho_2 = \lambda_2/(\mu c)$.

A. Estimating R_2

Let $n_q(t_1)$ be the average queue size at time $t < t_1$ and $n_q(t_2)$ be the queue size immediately before $t = t_2$. When the traffic intensity reaches its peak right after $t = t_1$, all c servers are busy almost 100% of the time, especially for $\rho_2 \gg 1$, and the rate at which the system completes work is μc, while the rate at which it receives work is λ_2. So, work accumulates at a rate of $\lambda_2 - \mu c$ requests/sec. Therefore,

$$n_q(t_2) = n_q(t_1) + (\lambda_2 - \mu c)\beta \tag{1}$$

where β is the surge duration.

But, because $\rho_2 > 1$ and $\rho_1 < 1$, we can expect that the growth in queue size during the surge will be much higher than the queue size before the surge. Thus,

$$n_q(t_2) \approx (\lambda_2 - \mu c)\beta. \tag{2}$$

The peak in response time occurs for a request that arrives at approximately $t = t_2$ and finds the longest queue size, which is $n_q(t_2)$. We call this the *tagged* request. This request leaves the system at $t = t_2^{'}$ and its response time, $R(t_2^{'})$, is approximately equal to the time needed to serve all requests it finds in the queue plus its own service time $1/\mu$. When the system is constantly busy, requests complete at a rate of μc requests/sec. Therefore, it will take $n_q(t_2)/(\mu c)$ sec to serve all $n_q(t_2)$ requests. Thus,

$$\begin{aligned} R(t_2^{'}) &\approx \frac{1}{\mu} + \frac{n_q(t_2)}{\mu c} = \frac{1}{\mu}\left[1 + \frac{n_q(t_2)}{c}\right] \\ &= \frac{1}{\mu}\left[1 + \frac{(\lambda_2 - \mu c)\beta}{c}\right]. \end{aligned} \tag{3}$$

But, if $(\lambda_2 - \mu c)\beta \gg c$ (i.e., a heavy load assumption) we can write that

$$\begin{aligned} R(t_2^{'}) &\approx \frac{(\lambda_2 - \mu c)\beta}{\mu c} \\ &= (\rho_2 - 1)\beta. \end{aligned} \tag{4}$$

B. Estimating δ

The tagged request leaves the system at $t = t_2^{'}$. This request's response time, $R(t_2^{'}) = t_2^{'} - t_2 = \delta$. Thus, using Eq. (4), we obtain

$$\delta = R(t_2^{'}) \approx (\rho_2 - 1)\beta. \tag{5}$$

Because $t_2^{'} - t_2 = \delta$,

$$t_2^{'} = t_2 + \delta. \tag{6}$$

C. Estimating Δ

We estimate now the average time, $\Delta = t_3^{'} - t_2^{'}$, for the response time to return to its value before the surge. The requests that arrive after the tagged request but before that request leaves the system accumulate at a rate of λ_1 during the response time $R(t_2^{'})$ of the tagged request. After the tagged request leaves at time $t_2^{'}$ and before time $t_3^{'}$, an additional $\lambda_1\Delta$ requests accumulate in the system. Thus, the total number N of requests accumulated since t_2 (end of the surge) until $t_3^{'}$ (return to the normal response time) is

$$N = \lambda_1 R(t_2^{'}) + \lambda_1\Delta = \lambda_1[R(t_2^{'}) + \Delta] \tag{7}$$

Let X be the average rate at which the system serves requests. Thus the time it takes to serve all N requests is

$$\Delta = N/X = \frac{\lambda_1[R(t_2^{'}) + \Delta]}{X}. \tag{8}$$

An approximate value for Δ can be obtained by assuming that the number of requests in the system will always be at least equal to c, so the rate $X = \mu c$. Thus,

$$\begin{aligned} \Delta &\approx \frac{\lambda_1[R(t_2^{'}) + \Delta]}{\mu c} \\ &= \rho_1[R(t_2^{'}) + \Delta]. \end{aligned} \tag{9}$$

Solving Eq. (9) for Δ gives us

$$\Delta = t_3^{'} - t_2^{'} \approx \frac{\rho_1}{1 - \rho_1} R(t_2^{'}) \tag{10}$$

Using the value of $R(t_2^{'})$ from Eq. (4) in Eq. (10) yields

$$\Delta = t_3^{'} - t_2^{'} \approx \frac{\rho_1(\rho_2 - 1)}{1 - \rho_1}\beta. \tag{11}$$

So, the time for the response time to return to its steady state value after the surge ends is $\delta + \Delta$ according to Fig. 3. This value can be computed using Eqs. (5) and (11):

$$\begin{aligned} \delta + \Delta &\approx (\rho_2 - 1)\beta + \frac{\rho_1(\rho_2 - 1)}{1 - \rho_1}\beta \\ &= \frac{\beta(\rho_2 - 1)}{1 - \rho_1}. \end{aligned} \tag{12}$$

D. Estimating $\oplus R$

The rate of increase in the response time, $\oplus R$, can be estimated by computing the slope of the line in Fig. 3 that shows the increase in response time due to the surge. Thus,

$$\oplus R = \frac{R_2 - R_1}{\beta + \delta}. \tag{13}$$

But assuming that $R_2 \gg R_1$ and using Eqs. (4) and (5) we obtain

$$\oplus R \approx \frac{(\rho_2 - 1)\beta}{\beta + \beta(\rho_2 - 1)} = \frac{\rho_2 - 1}{\rho_2}. \tag{14}$$

It is interesting to note that this slope is independent of the duration of the surge and only depends on the traffic intensity during the surge.

E. Estimating $\ominus R$

The value of $\ominus R$ can be estimated as the negative slope of the response time line that indicates the return from the peak response time to its original value. So,

$$\ominus R = \frac{-R_2 + R_1}{\Delta}. \quad (15)$$

But, using the heavy load assumption and Eqs. (4) and (11) we obtain

$$\ominus R \approx \frac{-R_2}{\Delta} = \frac{-(\rho_2 - 1)\beta}{[\rho_1(\rho_2 - 1)\beta]/(1 - \rho_1)} = \frac{\rho_1 - 1}{\rho_1}. \quad (16)$$

This estimate neither depends on the surge duration nor on the peak response time; it only depends on the traffic intensity before the surge.

IV. Validation

This section describes how we validated the equations derived in the previous section. To that end we developed a simulator for a G/G/c queue. The simulator also includes the autonomic elasticity controller described in Section V.

A. The G/G/c Simulator

The G/G/c simulator (see Fig. 4) is a discrete-event simulator that generates request arrival events and service time completions events according to given distributions for a G/G/c system. For the generation of non-exponential distributions we used Coxian distributions and the procedure described in [14] for generating values that follow a distribution with a given mean and given coefficient of variation. Generated events are placed in a calendar of events in chronological order and are removed and processed by either a Queue Manager in the case of arrival events or the Multi-Server Manager in the case of service completion events. Statistics about requests (e.g., time in the queue, service time) are recorded by the Statistics Manager so that reports, including 95% confidence intervals, can be generated when the simulation is complete. The Simulation Control module controls the duration of the simulation which consists of 100 independent runs for each set of parameters. Time is divided into slots of duration equal to 2 sec for each run at the post-processing stage. The values of $R(t)$ are computed as the averages of the $R(t)$ values in the same slot across all runs. The Autonomic Elasticity Controller is activated at regular time intervals and makes a decision about increasing or decreasing the number of servers. Details about this controller are given in Section V.

To validate the implementation of the simulator we compared its results with those of well-known analytic steady-state results for queues such as D/D/1, D/D/c, M/M/1, M/M/c, M/G/1, M/G/c, G/G/1, and G/G/c. Some of these results are exact and some are approximations (e.g., G/G/1 and G/G/c). Our comparisons varied the values of the traffic intensity from 0.5 to 0.95 and varied the number of servers from 1 to 9. The absolute relative percentage error between simulation and steady state analytic results was very small (less than 1% in most cases where exact queuing results exist and less than 5% where only approximate results exist). The validated simulator

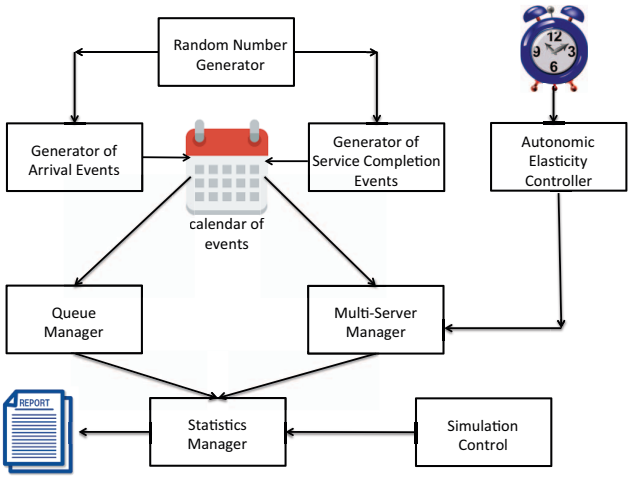

Fig. 4. Block Diagram of the G/G/c Simulator with the Autonomic Controller

was then used in situations where the offered load exceeds the system capacity. These are the cases for which we derived the analytical results in the previous section. A comparison between these results and simulation results is presented next.

B. Comparing Estimates with Simulation

Figure 5 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; $C_a = C_s = 2.0$. Therefore, $\rho_1 = 0.5$ and $\rho_2 = 2.0$. 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 285 sec, while the estimated value of R_2, according to Eq. (4), is $(\rho_2 - 1)\ \beta = (2 - 1) \times 300 = 300$ sec. The peak response time according to Fig. 5 occurs at time 900 sec. Since the surge ends at time 600 sec, $\delta = 900 - 600 = 300$ sec. This is precisely the value estimated by Eq. (5). The estimated value of Δ is, according to Eq. (11), $\rho_1(\rho_2 - 1) \times \beta/(1 - \rho_1) = 0.5(2 - 1) \times 300/(1 - 0.5) = 300$ sec. This estimated value would take us to time 900 sec + 300 sec = 1,200 sec, which is close to the 1,280 sec shown in the figure. The slope $\oplus R$ according to the figure is 285 / (900 - 300) = 0.475, which compares well with the value $(\rho_2 - 1)/\rho_2 = (2 - 1)/2 = 0.5$ predicted by Eq. (14). The response time decreases in an almost linear fashion most of the time after it reaches the peak. The slope of the linear portion of this line according to the figure is approximately - 285 / 1,200 = - 0.95 which matches closely with the estimated value of $(\rho_1 - 1)/\rho_1 = (0.5 - 1)/0.5 = -1$ predicted by Eq. (16).

We now show several other comparisons between estimated values and simulation results in Table I, which 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), and the values of $\delta, \Delta, \oplus R$, and $\ominus R$ obtained by

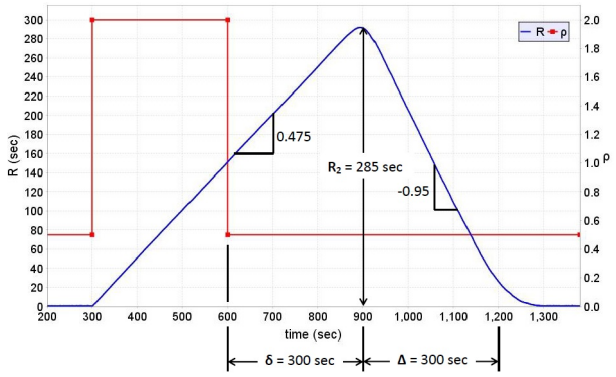

Fig. 5. Example of workload surge and corresponding effect on the response time for 5 servers (average over 100 runs). Workload surge duration: 300 sec; average service time: 0.5 sec; average arrival rate before and after surge: 5.0 requests/sec; average arrival rate during surge: 20.0 requests/sec; coefficient of variation of the interarrival time: 2; coefficient of variation of the service time: 2

the G/G/c simulator and through the estimates. The simulation values are averages over 100 runs and also show the 95% confidence intervals. The number of requests processed in each run varied between one million and 1.5 million. We also report the percent absolute error between the simulation and the analytic estimates as

$$\epsilon = 100 \times (\text{simulation} - \text{estimate})/\text{simulation}. \quad (17)$$

Experiments 1-6 are for G/G/5 and experiments 7 and 8 are for M/M/5 and D/D/5, respectively. The surge duration β 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 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 used for the coefficients of variation C_a, for the interarrival time, and C_s, for the service time distributions. Clearly, for M/M/5 $C_a = C_s = 1$ and for D/D/1 $C_a = C_s = 0$.

The first observation is that the percent error ϵ is very small (less than 8%) in all cases, i.e., for all experiments and all five metrics. The largest errors occur for Δ, in which case the maximum error is 7.6% and occurs in experiment 1. Even in that case, the predicted value of Δ, which is 45 sec, is very close to the lower bound (46.2 sec) of the 95% confidence interval. We also observe that as the surge duration β increases from 60 sec to 300 sec from experiment 1 to 5, the error ϵ for Δ tends to decrease. This is expected because of the heavy load assumption ($[\rho_2 - 1]\beta \gg c$) used in the derivation of the analytic predictions. For the same reason, higher values of ρ_2 improve the accuracy of the prediction. This can be seen in experiment 6 that uses $\rho_2 = 2.0$.

C. Analyzing the Effects of C_a and C_s

Here we examine the effects of C_a and C_s on the five metrics we proposed in Section III 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.

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 and keeping $c = 5$ for all experiments.

- $C_a \in \{0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0\}$; or 0.0 for $D/G/5$;
- $C_s \in \{0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0\}$; or 0.0 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 177 million for each model.

Table II shows the mean and 95% confidence intervals for the errors of the metrics for the three *sets* of experiments. In addition, Table III presents the 95% confidence intervals for a future value for all five metrics, i.e., the interval in which values obtained in an additional experiment would lie.

We summarize our observations below, which are based on (1) the statistical results presented in the tables and (2) individual experimental results not presented here due to space limitation.

- The mean absolute error ranged from 0.04% [for $G/D/5 : \epsilon(\delta)$] to 3.92% [for $G/G/5 : \epsilon(\Delta)$]. Hence, the analytic estimates are good proxies for the actual results of a *set* of experiments. Because the widths 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 β, ρ_2 and low values of C_a and C_s resulted in lower errors.
- Higher errors were obtained for combinations of input parameters with low values of β 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 β, $\rho_1 < 1.0$ 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 high single digit %, with the exception of $\epsilon(\delta)$ and $\epsilon(\Delta)$ for $G/G/5$.

Based on all the above observations one can conclude that the estimate formulas we proposed for the five 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.

TABLE I
COMPARISON BETWEEN ESTIMATES AND SIMULATION.

	[#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; $\beta = 60$ sec				
	R_2 (sec)	δ (sec)	Δ (sec)	$\oplus R$	$\ominus R$
Simulation	30.56 ± 1.0	30.37 ± 1.2	48.70 ± 2.5	0.340 ± 0.008	-0.660 ± 0.029
Estimated	30.00	30.00	45.00	0.333	-0.667
ϵ	1.8%	1.2%	7.6%	2.1%	-1.1%
	[#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; $\beta = 120$ sec				
	R_2	δ	Δ	$\oplus R$	$\ominus R$
Simulation	60.90 ± 1.6	60.13 ± 1.6	94.06 ± 3.6	0.340 ± 0.006	-0.660 ± 0.020
Estimated	60.00	60.00	90.00	0.333	-0.667
ϵ	1.5%	0.2%	4.3%	2.1%	-1.1%
	[#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; $\beta = 180$ sec				
	R_2	δ	Δ	$\oplus R$	$\ominus R$
Simulation	90.01 ± 1.9	89.58 ± 2.0	139.72 ± 4.4	0.330 ± 0.005	-0.650 ± 0.016
Estimated	90.00	90.00	135.00	0.333	-0.667
ϵ	0.0%	-0.5%	3.4%	-0.9%	-2.6%
	[#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; $\beta = 240$ sec				
	R_2	δ	Δ	$\oplus R$	$\ominus R$
Simulation	120.45 ± 2.0	120.10 ± 2.1	183.98 ± 5.3	0.330 ± 0.004	-0.660 ± 0.016
Estimated	120.00	120.00	180.00	0.333	-0.667
ϵ	0.4%	0.1%	2.2%	-0.9%	-1.1%
	[#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; $\beta = 300$ sec				
	R_2	δ	Δ	$\oplus R$	$\ominus R$
Simulation	151.36 ± 2.5	151.07 ± 2.6	231.26 ± 6.5	0.340 ± 0.004	-0.660 ± 0.014
Estimated	150.00	150.00	225.00	0.333	-0.667
ϵ	0.9%	0.7%	2.7%	2.1%	-1.1%
	[#6] $G/G/5$; $C_a = 1.5$; $C_s = 1.5$; $\rho_1 = 0.6$; $\rho_2 = 2.0$; $R_1 = 0.633$ sec; $\beta = 300$ sec				
	R_2	δ	Δ	$\oplus R$	$\ominus R$
Simulation	300.54 ± 3.3	301.24 ± 3.4	449.90 ± 9.9	0.500 ± 0.003	-0.670 ± 0.014
Estimated	300.00	300.00	450.00	0.500	-0.667
ϵ	0.2%	0.4%	-0.0%	0.0%	0.4%
	[#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; $\beta = 300$ sec				
	R_2	δ	Δ	$\oplus R$	$\ominus R$
Simulation	150.22 ± 1.9	149.97 ± 2.0	230.64 ± 4.5	0.330 ± 0.003	-0.650 ± 0.009
Estimated	150.00	150.00	225.00	0.333	-0.667
ϵ	0.1%	-0.0%	2.4%	-0.9%	-2.6%
	[#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; $\beta = 300$ sec				
	R_2	δ	Δ	$\oplus R$	$\ominus R$
Simulation	150.03 ± 0.0	151.00 ± 0.0	226.00 ± 0.0	0.330 ± 0.000	-0.660 ± 0.000
Estimated	150.00	150.00	225.00	0.333	-0.667
ϵ	0.0%	0.7%	0.4%	-0.9%	-1.1%

TABLE II
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	$\epsilon(R_2)$	1.25% ± 0.3%	0.80% ± 0.3%	1.43% ± 0.4%
2	$\epsilon(\delta)$	0.04% ± 0.5%	-0.41% ± 0.4%	0.14% ± 1.2%
3	$\epsilon(\Delta)$	3.24% ± 0.7%	3.23% ± 0.6%	3.92% ± 0.8%
4	$\epsilon(\oplus R)$	0.89% ± 0.3%	0.58% ± 0.3%	1.04% ± 0.5%
5	$\epsilon(\ominus R)$	-1.11% ± 0.4%	-1.55% ± 0.4%	-0.37% ± 0.5%

TABLE III
95% CONFIDENCE INTERVALS FOR A FUTURE VALUE OF THE ERRORS FOR THE METRIC ESTIMATES

	ϵ (metric)	$G/D/5$	$D/G/5$	$G/G/5$
1	$\epsilon(R_2)$	1.25% ± 3.5%	0.80% ± 3.0%	1.43% ± 5.3%
2	$\epsilon(\delta)$	0.04% ± 6.3%	-0.41% ± 4.4%	0.14% ± 14.6%
3	$\epsilon(\Delta)$	3.24% ± 7.8%	3.23% ± 7.2%	3.92% ± 9.9%
4	$\epsilon(\oplus R)$	0.89% ± 3.8%	0.58% ± 3.1%	1.04% ± 5.8%
5	$\epsilon(\ominus R)$	-1.11% ± 5.0%	-1.55% ± 4.7%	-0.37% ± 6.0%

V. 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 the servers are virtualized, it is relatively easy to change the number of virtual machines and/or their characteristics.

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 β 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 equation (see Eq. (4)) to estimate the minimum number of servers, $c_{\min}$, needed to maintain the peak response time below $R_{\max}$. Thus,

$$R_2 \quad \approx \quad (\rho_2 - 1)\beta \leq R_{\max} \Rightarrow \rho_2 \leq \frac{R_{\max}}{\beta} + 1$$

$$\Rightarrow \quad \frac{\lambda_2}{\mu c} \leq \frac{R_{\max}}{\beta} + 1$$
$$\Rightarrow \quad c \geq \frac{\lambda/\mu}{\frac{R_{\max}}{\beta} + 1}. \tag{18}$$

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

$$c_{\min} = \left\lceil \frac{\lambda/\mu}{\frac{R_{\max}}{\beta} + 1} \right\rceil. \tag{19}$$

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 what we did in the equation above we get,

$$\mu_{\min} = \frac{\lambda/c}{\frac{R_{\max}}{\beta} + 1}. \tag{20}$$

If the server capacities can only be selected from a discrete set such as in the set of VMs available from Amazon EC2, then the value of $\mu_{\min}$ can be used the determine the lowest capacity VM that is faster 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 of original servers, and (4) the maximum desirable response time $R_{\max}$. The controller wakes up at regular intervals (called controller intervals), checks if the traffic intensity is below a surge level ($\rho < 1$; see lines 5-8) or experiencing a surge ($\rho \geq \rho_{\text{original}}$; see lines 10-19). The pseudo-code in Algorithm 1 assumes that λ always has the value of 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.

While the system is experiencing a surge, the surge duration β is updated at line 12 each time the loop in lines 10-19 is executed. Because the controller sleeps for τ seconds within the loop, β is incremented by τ every time. In line 13, the minimum number of servers $c_{\min}$ is recomputed with the updated value of β. As β increases, $c_{\min}$ 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

```
   Input : λ, μ, c_original, R_max
1  ρ_original ← λ/(μ c_original)
2  while system is online do
3      c ← c_original
4      ρ ← λ/(μ c)
       /* wake up at regular intervals
          while surge has not started      */
5      while ρ < 1 do
6          Sleep (τ)
7          ρ ← λ/(μc)
8      end
9      SurgeStart ← CurrentTime - τ/2
       /* While surge is ongoing            */
10     while ρ ≥ ρ_original do
11         ρ ← λ/(μ c)
           /* Update surge duration         */
12         β ← CurrentTime - SurgeStart
           /* Compute min. # of servers     */
13         c_min = ⌈ λ/μ / ((R_max/β)+1) ⌉
14         if c_min > c then
               /* Change # required servers */
15             Change # servers in the system to c_min
16             c ← c_min
17         end
18         Sleep (τ)
19     end
       /* return to original # servers      */
20     Change # servers in the system to c_original
21 end
```

Due to space limitations we do not show results here for the vertical elasticity controller.

VI. Simulation-based Evaluation of the Controller

This section presents an experimental evaluation of the autonomic elasticity controller discussed in the previous section. We define here some additional metrics we used to assess the effectiveness of the controller:

- D: time during which the response time suffered; $D = t_3^{'} - t_1 = \beta + \delta + \Delta$.
- Φ: area under the curve of $R(t)$ during D; more specifically, $\Phi = \int_{t_1}^{t_1+D} R(t)dt$.

These metrics are identified in Fig. 6, which shows the behavior of a multi-server system during a workload intensity surge, with and without the controller. We use the subscripts or superscripts c and nc heretofore 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:

$$\eta(X) = 100 \times \frac{X_{nc} - X_c}{X_c}. \tag{21}$$

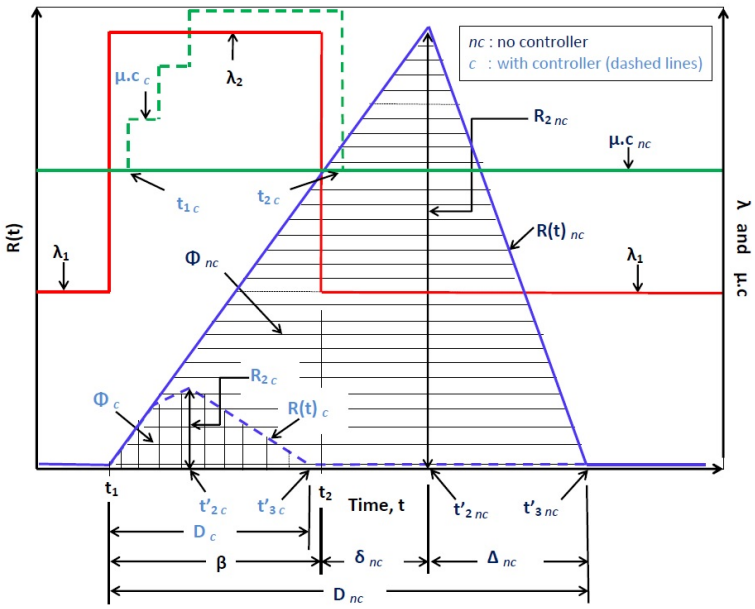

Fig. 6. Comparing surge behavior with and without a controller.

Algorithm 2: Vertical Elasticity Controller

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 (τ)
    $\rho \leftarrow \lambda/(\mu c)$
  end
  SurgeStart ← CurrentTime - $\tau/2$
  /* While surge is ongoing */
  while $\rho \geq \rho_{\text{original}}$ **do**
    $\rho \leftarrow \lambda/(\mu\ c)$
    /* Update surge duration */
    $\beta \leftarrow$ CurrentTime - SurgeStart
    /* Compute min. server capacity */
    $\mu_{\min} = \frac{\lambda/c}{(R_{\max}/\beta)+1}$
    if $\mu_{\min} > \mu$ **then**
      /* Update server capacity */
      Change server capacity to $\mu_{\min}$
      $\mu \leftarrow \mu_{\min}$
    end
    Sleep (τ)
  end
  /* return to original capacity */
  Change server capacity to μ_{original}
end

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.

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. 6:

$$R_{2_{nc}} = (\rho_2 - 1)\beta \quad (22)$$

$$D_{nc} = \beta + \delta_{nc} + \Delta_{nc} \quad (23)$$

$$\Phi_{nc} = \frac{1}{2} R_{2_{nc}}.D_{nc} = \frac{1}{2}.\beta.(\rho_2 - 1).(\beta + \delta_{nc} + \Delta_{nc}) \quad (24)$$

Note that Φ_{nc} is obtained by computing the area of the triangle with height $R_{2_{nc}}$ and base D_{nc}. This is a slightly conservative approximation (i.e., a slightly lower value) of Φ_{nc}. Therefore, according to Eq. (21), $\eta(\Phi)$ is slightly conservative, which is a good property.

We conducted experiments with the horizontal elasticity controller and measured the three metrics described above and computed the improvement η as reported in Table V. The no controller values in the table are obtained from Eqs. (22)-(24). 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 of 300 sec. The first general observation is that the percent relative improvement $\eta()$ is very large (ranges from 53 to 2,901) 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. Finally, for the three experiments with $\beta = 300$ sec, the G/G/c one has higher gains for D and Φ than the other two.

TABLE IV
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; $R_{max} = 5$ sec; $\beta = 60$ sec				
		No controller	With Controller	$\eta()\%$
1	R_2 (sec)	30.0	13.5	122.3
2	D (sec)	135.0	88.3	53.0
3	Φ (sec^2)	2,025.0	745.1	171.8
[b] $G/G/c$ with the same parameters as above and $\beta = 180$ sec				
		No controller	With Controller	$\eta()\%$
1	R_2 (sec)	90.0	14.2	536.0
2	D (sec)	405.0	169	140
3	Φ (sec^2)	18,225	1,543	1,081
[c] $G/G/c$ with the same parameters as above and $\beta = 300$ sec				
		No controller	With Controller	$\eta()\%$
1	R_2 (sec)	150.0	14.7	921
2	D (sec)	675.0	189	257
3	Φ (sec^2)	50,625	1,687	2,901
[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; $R_{max} = 5$ sec; $\beta = 300$ sec				
		No controller	With Controller	$\eta()\%$
1	R_2 (sec)	150.0	14.2	957
2	D (sec)	675.0	218	210
3	Φ (sec^2)	50,625	1,906	2,556
[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; $R_{max} = 5$ sec; $\beta = 300$ sec				
		No controller	With Controller	$\eta()\%$
1	R_2 (sec)	150.0	14.4	941
2	D (sec)	675.0	299.0	126
3	Φ (sec^2)	50,625.0	2,476	1,944

The behavior of the controller can be appreciated in Fig. 7 that shows 10 plots that illustrate (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, the value of β varies from 60 (top) to 300 (bottom) sec in increments of 60 sec. The controller interval was set to $\tau = 15$ sec in all cases.

The following observations can be drawn from the plots in Fig. 7:

- In all 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 $\beta = 60$ sec and $R_{\max} = 10$ sec, the controller brought the peak in response time to about a half of the value without the controller. But, for $\beta = 300$ sec, the peak response time is reduced to about 13% of the value without the controller. This is the case because for larger surge durations the controller has more possibilities to adjust the number of servers. For example, for $\beta = 300$ sec the controller made more adjustments to the number of servers while for $\beta = 60$ sec fewer adjustments were made.
- The peak response time with the controller was higher than the maximum desired value because we selected a controller interval τ value (15 sec) relatively large with respect to the surge durations used. For example, for the small surge of 60 sec the controller would only be able to act 4 times. Smaller values for τ would bring the peak response time to lower values. It should be noted however that there usually is a delay involved in provisioning more resources (e.g., launching more virtual machines). Thus, it may be useful for the controller to do some minor overprovisioning above $c_{\min}$ to counter the negative effects on performance due to long VM launch times [15].
- The controller tries to minimize the cost of using more servers by incrementally adding more servers as it has a better estimate of the surge duration.

As Fig. 7 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:

$$\kappa = \int_{t_{1_c}}^{t_{2_c}} [c_c(t) - c_{nc}] \ \ dt \tag{25}$$

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 V 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 values of the surge duration β were used and two values of $R_{\max}$ were used for two different values of the controller interval time τ. The following interesting observations can be gleaned from the table: (1) as the surge duration increases so does the additional server-seconds, κ, 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 frequently responsive controller ($\tau = 15$ vs. $\tau = 30$) provides more resources to counter the surge.

We also conducted experiments aimed at demonstrating the robustness of the controller even when the workload does not have the exact shape assumed for the derivation of the equations used by the controller. For that purpose, we injected random upward and downward perturbations every τ seconds to the average workload intensity outside and during the surge. These modified average workload intensity values drive the

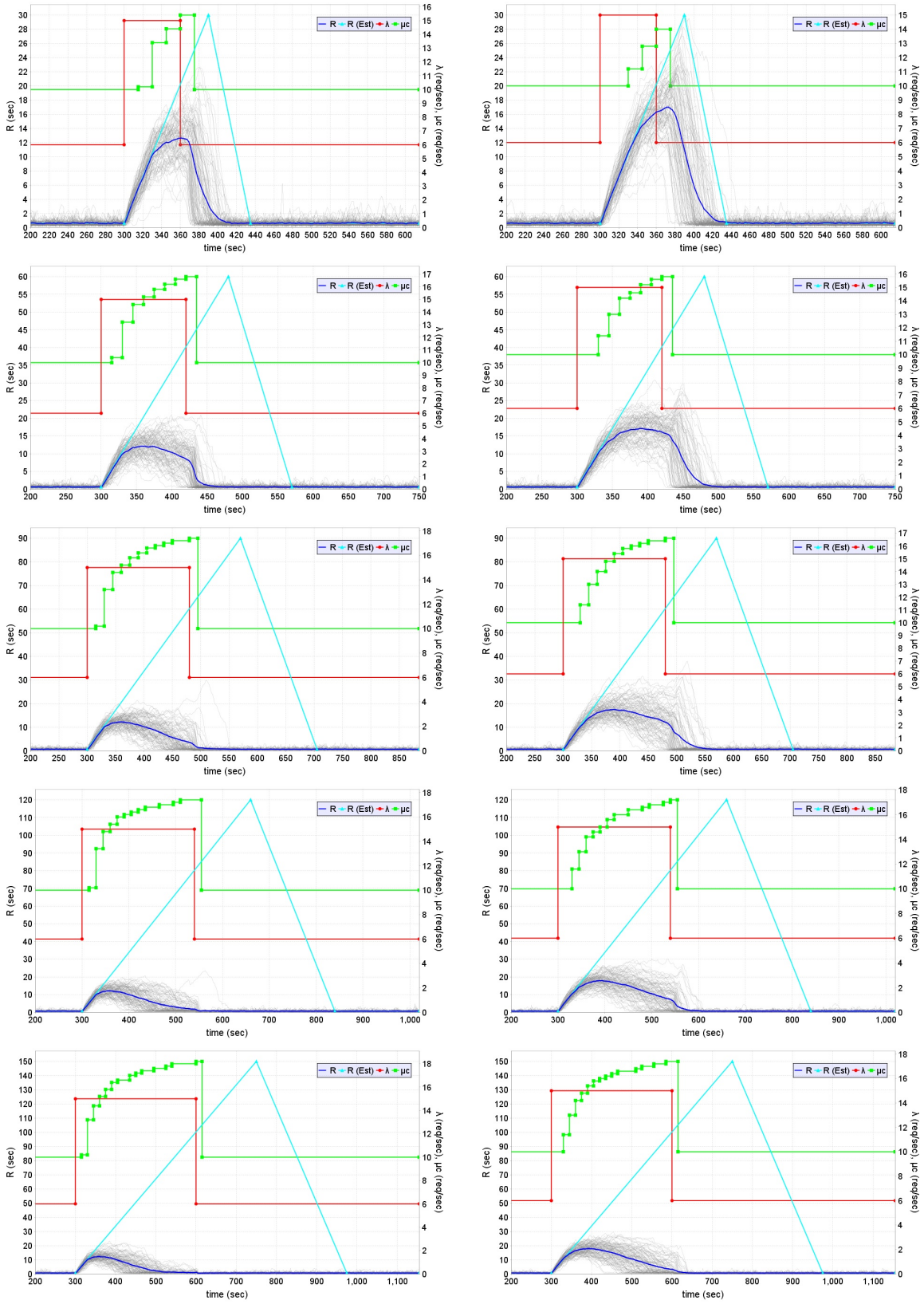

Fig. 7. With Horizontal Elasticity Controller - $G/G/5; C_a = 1.40; C_s = 1.30; \rho_{1_{nc}} = 0.60; \rho_{2_{nc}} = 1.50; \beta = 60, 120, 180, 240, 300; \tau = 15$ sec; $R_{\max} = \{$left column= 5 sec; right column= 10 sec$\}$; Average over 100 independent runs.

TABLE V
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				
	$\tau = 15$ sec		$\tau = 30$ sec	
β (sec)	$R_{\max} = 5$	$R_{max} = 10$	$R_{\max} = 5$	$R_{\max} = 10$
60	99	65	102	51
120	291	219	267	195
180	509	411	462	366
240	713	645	651	555
300	962	846	852	735

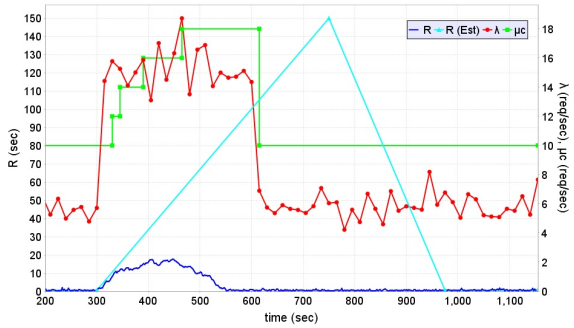

Fig. 8. Controller effect under a 5% perturbation of the average arrival rate.

generation of the interarrival times during each interval of τ seconds. For example, Fig. 8 shows that the controller correctly detects the onset of the surge, increases the number of servers five times during the surge, and decreases the number of servers to its original value when it detects the end of the surge.

VII. RELATED WORK

The authors of [16] conducted a comprehensive study on existing 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 [17] describes a broker to acquire resources on demand from a public cloud to service requests from a client enterprise. That work does not present any equations that can be used to predict the effects of workload intensity surges. The authors of [18] presented an elasticity management framework that takes the input typically presented to reactive rule-based scaling strategies and return a proactive auto scaler. Examples of predictive control can be found in [19], [20]. The authors of [21] 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 involved in autoscaling in the cloud including the need to accurately estimate resource usage in the face of significant variability in workload patterns. The authors of [22] discussed such challenges and then presented a model-predictive algorithm for workload forecasting that is used for resource autoscaling. The sensitivity of auto-scaling mechanisms to the prediction results has been investigated by Nikravesh et al. [23]. Their work compared threshold-based scaling techniques based on Support Vector Machine (SVM) and Neural Networks (NN) predictions. The authors of [24] provided an extensive review of the two broad categories of autoscaling techniques—reactive and proactive—and discuss in detail five different groups: static, threshold-based policies, reinforcement learning, queuing theory, control theory, and time-series analysis. The authors of [25] 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 authors of [26] 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 [27] 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 authors in [28] have 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 [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.

VIII. CONCLUDING REMARKS AND FUTURE WORK

Many computing environments, such as web sites with multiple web servers, use many web servers to serve requests that arrive from a multitude of customers. Such systems may be subject to surges in the traffic intensity such that during intervals of time the offered load exceeds the system's capacity. This paper derived equations that can be used to estimate the impact of a surge on the peak response time caused by the surge, on the time lag between the end of the surge and the time at which the response time peaks, on the rate at which the response time grows due to the surge and on the rate in which it decreases after it reached a peak. These equations were extensively validated using a simulator we developed, which showed that the error between the analytic estimates and simulation is very small in all cases analyzed.

We then designed and implemented two proactive 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. It should be noted that 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 designing a hybrid elasticity controller that combines horizontal with vertical elasticity. Second, we are in the process of analyzing the Google traces [30] 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 more general surge patterns such as trapezoid ones. However, we showed in Fig. 8 that our controller works well even when the workload surge is not exactly rectangular. We feel that our work is an important first step towards generalizing the model-driven controller given that many practical workload surges may be synthesized from a combination of rectangular surges and other simple patterns. Additionally, we intend to design controllers that use the surge characterization to predict the surge duration if the surge duration has exceeded a certain value.

ACKNOWLEDGEMENTS

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

REFERENCES

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

[2] H. Kobayashi and B. L. Mark, *System Modeling and Analysis: Foundations of System Performance Evaluation*. Prentice Hall, 2009.

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

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

[11] 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.

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

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

[14] M. Bennani and D. Menascé, "Assessing the robustness of self-managing computer systems under highly variable workloads," in *Autonomic Computing, 2004. Proc. Intl. Conf.* IEEE, 2004, pp. 62–69.

[15] 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.

[16] 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.

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

[18] 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.

[19] 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.

[20] G. Mencagli, M. Vanneschi, and E. Vespa, "A cooperative predictive control approach to improve the reconfiguration stability of adaptive distributed parallel applications," *ACM Trans. Auton. Adapt. Syst.*, vol. 9, no. 1, pp. 2:1–2:27, Mar. 2014.

[21] 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.

[22] 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.

[23] 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 Conference Workshops*, July 2014, pp. 690–695.

[24] 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.

[25] 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.

[26] 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

[27] A. Bauer, N. Herbst, and S. Kounev, "Design and evaluation of a proactive, application-aware auto-scaler: Tutorial paper," in *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering*, ser. ICPE '17. New York, NY, USA: ACM, 2017, pp. 425–428. [Online]. Available: http://doi.acm.org/10.1145/3030207.3053678

[28] 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. [Online]. Available: http://doi.acm.org/10.1145/3030207.3030227

[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] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: format+ schema," *Google Inc., Technical Report*, pp. 1–14, Nov 2011. [Online]. Available: http://code.google.com/p/googleclusterdata/wiki/TraceVersion2