Improving Predictability of User-Affecting Metrics to Support Anomaly Detection in Cloud Services

Alberto Avritzer*, Vilc Rufino¶, Daniel Menasché¶, Barbara Russo‡, Andrea Janes‡,
Vincenzo Ferme§, André van Hoorn§ and Henning Schulz§
*eSulab Solutions, Princeton, USA, ¶Federal University of Rio de Janeiro (UFRJ), Rio de Janeiro, Brazil
‡Free University of Bozen-Bolzano, Bolzano, Italy, §University of Stuttgart, Stuttgart, Germany

***Abstract*—Intrusion detection systems (IDSs) aim for detecting and reporting security attacks to systems or networks. Previous work has shown that intrusions have an impact on system performance, and that performance signatures can be used effectively for implementing an IDS. In this paper, we present an analytical and an experimental study on the tradeoff between intrusion detection based on performance signatures and system scalability. The proposed approach combines analytical modeling and load testing to find optimal configurations for the signature-based IDS. We apply a heavy-tail bi-modal modeling approach, where "short" jobs represent normal transactions and "long" jobs represent denial of service attacks. Then, we parameterize the bi-modal model using results obtained from load testing and estimate the baseline performance average and standard deviation for several system parameter values (e.g., different ratios of "short" and "long" jobs).**

Using the proposed model, we discovered a fundamental tradeoff between performance and security. The key insights from our analysis are: (i) there is an optimal number of servers which minimizes the response time variance, (ii) the sweet-spot number of servers that minimizes response time variance is typically smaller than or equal to the one that minimizes the mean response time. Therefore, for security purposes, it may be worth slightly sacrificing mean response time to reduce response time variance.

I. Introduction

Background The multi-server paradigm (e.g., cloud computing provided by Amazon and Google) with a central queueing system (e.g., load balancing front end) has been used as the state of the practice for scalable server-side computation for the last several years [1], [2]. The setup of such systems, in turn, poses a number of challenges related to the shaping of the workload and the configuration of the servers. Figure 1 shows the interplay between anomaly detection, load balancing and system configuration (in terms of the number of servers) in an abstract representation of a cloud computing environment. For example, in [2], an architecture using a single queue load-balancing front-end combined with admission control was proposed to support predictable performance in cloud applications. The authors have shown that their approach was able to produce smaller response time variability than other evaluated strategies. More predictable response time, in turn, translates into higher security levels, as it is easier to tune anomaly detection algorithms when the target system is more predictable. If response time variance is high, for instance, it may be difficult to distinguish between transient performance degradation and a denial of service attack. In general, the set-up of scalable cloud computing services impacts both performance and system security aspects.

Goal In this paper, we develop a framework for the analysis of the fundamental trade-offs between security and performance/scalability in central-queue multi-server systems [3], [4]. Specifically, we are interested in analyzing the impact of deployment scale (e.g., number of servers) on the effectiveness of anomaly detection approaches based on performance signatures, as for example, response time. The proposed approach relies on the response time being predictable. Therefore, in this paper, we investigate the impact of the number of servers on the response time variability.

Challenges We focus on metrics directly affecting customers (e.g., response times), showing that they are sensitive to security intrusions to the extent that they can then be used to support an intrusion detection framework. As those metrics are intrinsically publicly available, sharing them with third-parties responsible for intrusion detection has the benefit of not compromising user privacy. This should be contrasted against deep packet inspection, for instance, which is more computationally expensive and requires the sharing of sensible data.

The task of an intrusion detection system (IDS) is the detection and alerting of malicious activities in computer systems or networks. Various IDS approaches have been proposed in the past. They can be classified based on the monitored platform (host, network, hybrid), the attack detection method (misuse-based, anomaly-based, hybrid), and the deployment architecture (non-distributed, distributed) [5]. Milenkoski et al. [5] report that one of the most important open research questions in this domain is the development of IDSs for detecting zero-

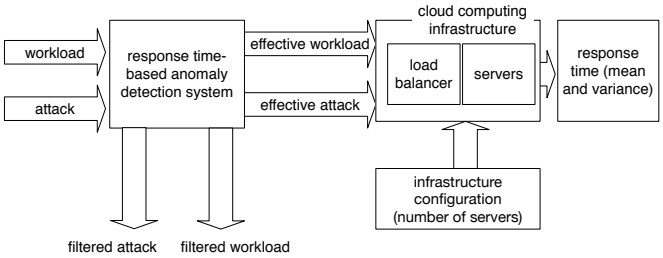


Fig. 1: Anomaly detection framework, accounting for infrastructure configuration actions at system deployment time.

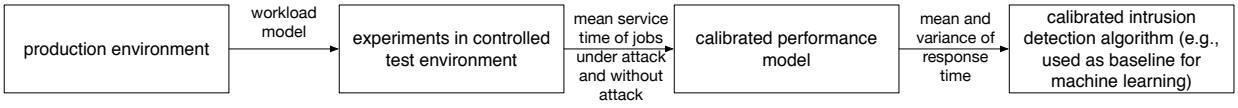


Fig. 2: Proposed framework for a performance-based anomaly detection system

day attacks and advanced persistent threats (APTs) [6]. In this context, anomaly detection approaches based on performance signatures have the potential of detecting zero-day attacks, as anomaly detection approaches are based on detecting performance deviations from normal behavior, and therefore do not require knowledge of attack history.

Prior art Previous work has shown that performance engineering can be used for implementing attack detection methods. For instance, in [7], an approach that used performance signatures based on CPU, I/O, memory and network usage, for the detection of security intrusions was introduced. The approach has been shown to be very accurate and to outperform, in some cases, intrusion detection systems [5] based on security intrusion history.

The architecture of performance and security in cloud computing systems incurs several tradeoffs, as for example, the overhead of implementing security protocols [8], the configuration of database partitioning systems for intrusion detection while optimizing performance [9], and security related cache invalidation analysis to ensure scalability of data intensive applications [10]. In contrast, in this paper, we evaluate the impact of cloud computing infrastructure configuration, e.g., number of servers allocation, on the application mean and variance of response time. If response time is to be used as a metric to support anomaly detection, it is important to setup the system such that response time variability can be attributed to security intrusions and not to the variability in system performance.

Methodology and contribution We apply a heavy-tail bimodal modeling approach, where "short" jobs represent normal transactions and "long" jobs represent denial of service attacks. In our modeling approach, "long" jobs are related to security intrusions. Using the bimodal modeling approach we are able to estimate the baseline system response time average and standard deviation, for several system parameter values (e.g., different ratios of "short" and "long" jobs).

Using the proposed model, we consider the problem of determining the optimal number of servers accounting for scalability and security aspects, e.g., intrusion detection based on average response time. We discovered that while the economics of scalability favors a larger number of servers, which reduces average service times, intrusion detection favors a reduction in the number of servers, to reduce standard deviation of service times. Figure 2 illustrates the framework using the concepts studied in this paper and is further discussed in the upcoming section.

Paper outline The outline of the remainder of the paper is as follows. In Section II, we present a performance-based anomaly detection framework that is composed of (i) workload modeling, (ii) empirical experiments, (iii) calibrated performance models and (iv) anomaly detection algorithms based on tracking of customer-affecting metrics, as for example, response time. In Section III, we describe the proposed analytic model that is used to estimate average and standard deviation of response time accounting for number of servers and security intrusions. In Section IV, we report on the results from controlled experiments executed on a testbed, and indicate how to parametrize the analytical model. In Section V, we present insights derived from the analysis of the numerical results obtained from the analytic model, which was parametrized using the results from the controlled testbed experiments. The broad implications of our results are discussed in Section VI. In Section VII, we present an overview of the related work followed by conclusions and directions for future research in Section VIII.

II. Proposed Framework

In this section, we describe the proposed performance-based anomaly detection framework introduced in this paper. Figure 2 shows the building blocks that compose the framework and the inputs and outputs of each process block, as follows:

1) The **production environment** is monitored using application performance monitoring (APM) tools [11], e.g., Influxdb[1] and Splunk[2] to generate a workload model, e.g., user profile distribution [12]. The obtained workload model is used to support the specification of the load levels to be used in the performance testing experiments.
2) **Experiments in controlled test environment** are executed to estimate the mean service time of jobs under attack and without attack. The load tests are executed using load levels specified by using the derived workload model. Laboratory experiments are run using load testing tools [13] to measure mean and standard deviation of service time, with and without intrusions. The submitted transaction rate is derived from APM tools. Experimental results to calibrate the performance model are presented in Section IV.
3) **Calibrate the analytic performance model** to assess the dependency of the mean and standard deviation of response times with respect to the number of servers. The calibration leverages experimental results reported in Section IV to characterize a bi-modal distribution. Short jobs correspond to service times of typical users, and long jobs correspond to security attacks. The analytic model results (reported in Section V) are used to estimate the number of servers that minimize average response time variance, as shown in Table III.

[1]https://www.influxdata.com/

[2]https://splunkbase.splunk.com/app/3065/

TABLE I: Table of notation

Variable	Description	Comment
ρ	utilization	$\rho = \rho_S + \rho_L = \lambda E(X)$
α	fraction of normal jobs (short)	
λ	arrival rate of jobs	
$E(X)$	mean service time	
T	metric of interest: response time	

4) **Calibrate the anomaly detection algorithms** using the mean and standard deviation of response time for the optimal number of servers. One alternative is to apply the Bucket algorithm introduced in [14]. Alternatively, machine learning algorithms, such as [15] can also be used. The calibration of anomaly detection algorithms using the proposed framework is out of the scope of this paper, and is left as subject for future work.

III. Analytical Model

In this section, we describe the proposed analytical model, which extends results by Psounis *et al.* [1]. The model aims at determining the impact of different system characteristics on the average and standard deviation of the response time. In particular, we account for the impact of the number of servers, the service time of normal jobs, and the service time of jobs serviced during an attack period.

A. Model description

We use an $M/G/K$ model to characterize our system [4], [3]. Under the $M/G/K$ model, arrivals occur according to a Poisson process to a queue served by K servers. When a service completes, the server starts working on the next job at the head of the line. Although the $M/G/K$ is one of the simplest models to capture the essence of a FIFO multi-server cloud service, most of its performance metrics are still not known in closed form, and understanding their behavior remains an open problem (see [3, Figure 1.7], [16] and [17, Section 7.4]).

Requests arrive according to a Poisson process with rate λ. We denote by $E(X)$ the mean service time of requests, which correspond to typical user requests or attacks. The mean service times are given by $E(X_S)$ and $E(X_L)$, respectively, where $E(X_S) \leq E(X_L)$. Figure 3 summarizes the key elements of the model. Table I summarizes the model parameters and notation.

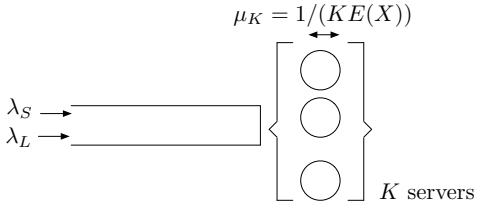


Fig. 3: $M/G/K$ model: the proposed analytical model is used to assess the impact of the number of servers K on standard deviation and mean response times (Section V).

Let λ_S and λ_L correspond to the arrival rate of normal jobs (short jobs) and attacks (long jobs), respectively. Let α be the fraction of requests corresponding to normal traffic,

$$\lambda = \lambda_S + \lambda_L, \quad \alpha = \lambda_S/\lambda. \tag{1}$$

Then,

$$E(X) = \alpha E(X_S) + (1-\alpha)E(X_L) \tag{2}$$

We denote by ρ the utilization of each server, and by ρ_S and ρ_L the utilization due to short and long jobs. Then,

$$\rho = \rho_S + \rho_L, \quad \rho_S = \lambda_S E(X_S), \quad \rho_L = \lambda_L E(X_L). \tag{3}$$

Let K be the number of servers in the system. We use the $M/G/K$ queue to model the number of requests in the system, and to characterize response times (request sojourn times). We denote by μ the mean service capacity (service rate) of a single server system, measured in jobs served per time unit,

$$\mu = 1/E(X). \tag{4}$$

Let μ_K denote the mean service capacity of each server when there are K servers in the system.

a) Scaling law: We assume that the total system capacity is uniformly distributed among servers,

$$\mu_K = 1/(KE(X)). \tag{5}$$

The service capacity of each of the K servers, μ_K, decreases as the number of servers increases, in such a way that the mean service capacity remains constant and the mean service time per server is increased to $KE(X)$.

b) Response time: We denote by T the system response time, our key metric of interest. We leverage results by Psounis *et al.* [1] to approximate $E(T)$. In essence, $E(T)$ is comprised of two components: (i) the time in the server, $KE(X)$, and (ii) for *blocked* jobs that have to wait in line, the waiting time which is approximated as the waiting time of an $M/G/1$ queue, given by $\rho E(X_r)/(1-\rho)$, where $E(X_r)$ is the residual service time. Then,

$$E(T) \approx KE(X) + \frac{\rho}{1-\rho}\frac{E(X^2)}{2E(X)}p_B(K,\rho_S,\rho_L), \tag{6}$$

where p_B is the *blocking* probability, $p_B(K,\rho_S,\rho_L) = (1 - F(K(1-\rho_S)-1))$ and $F(x)$ is the cumulative density function (CDF) of the Poisson distribution with mean $K\rho_L$ at point $\lfloor x \rfloor$. We refer the reader to [1] for further details, including the derivation of the blocking probability. Similarly,

$$E(T^2) \approx K^2E(X^2) + \frac{\rho}{1-\rho}\frac{E(X^3)}{3E(X)}p_B(K,\rho_S,\rho_L). \tag{7}$$

Throughout this paper, for our analytical results we assume that service times of short and long jobs are roughly deterministic: $E(X^i) \approx \alpha E(X_S)^i + (1-\alpha)E(X_L)^i$.

c) Additional simplifying approximations and simulations: In addition, to show that even a very simplified approximation of the model solution already allows us to appreciate the essence of our results, we also consider the following two simplifying assumptions when numerically assessing the model: 1) ρ_S small: if $\rho_S \approx 0$, then $\lfloor K(1-\rho_S)-1 \rfloor \approx K-2$; 2) approximate variance: as argued by Psounis *et al.* [1], if the service time of jobs under attack is very long compared to service time of short jobs, we also have that $V(T) \approx E(T^2)$ and $\sigma(T) = \sqrt{V(T)}$. Those simplifying assumptions serve to produce plots that are easy to visualize without noise that is inherent to simulations or more refined approximations (contrast Figures 4 and 5 produced with the approximations above against simulation results shown in Figure 6). We conducted a simulation campaign to validate the adopted simplifications, and verified that they serve to extract the essence of our results (Section V-D and [18]).

d) Model summary: In this section we have introduced the proposed analytical model: an $M/G/K$ queue subject to short and long jobs. Although the $M/G/K$ is very simple, there are no known closed form expressions for the moments of its response time [17], motivating simplifying approximations as suggested by [1], [19]. In the following sections, we consider the model in its simplest form, and indicate the insights that it entails with respect to system predictability and security. We verified that all the insights still hold without accounting for the simplifying approximations. Nonetheless, the simplified model facilitates visualization and reproducibility of results, and for this reason we focus on it. The source code to reproduce our results is available at [18].

It is quite remarkable that a simple and classic model such as the $M/G/K$ still allow us to derive fundamentally novel insights on the tradeoff between predictability and performance (see Section III-C). The insights derived from such simple model already serve to guide practitioners, as indicated in Section VI.

B. Optimal number of servers

The analytical model can be used to determine the impact of different system parameters on the mean and standard deviation of response times. In particular, it can be used to assess how K, ρ, α, $E(X_L)$ and $E(X_S)$ impact $V(T)$ and $E(T)$.

The optimal number of servers is determined by the value of K that minimizes a weighted sum of $V(T)$ and $E(T)$. Table II summarizes the key differences between a performance perspective towards finding the optimal value of K and a joint performance-security perspective, wherein both $V(T)$ and $E(T)$ must be taken into account. In essence, a performance perspective should focus on the average response time. A security perspective, in contrast, must also account for the response time standard deviation, as it is key for anomaly detection systems to distinguish between normal system performance and system performance under attacks.

To determine the optimal number of servers, we vary the number of servers K and solve the model for different values of K. Numerically assessing the impact of K on the metrics of interest allows us to find a value that trades between possibly conflicting goals, as show in Section V. In Section IV we describe the model parametrization from the controlled experiments.

C. Key insights

Next, we briefly report the key insights derived from the proposed model. They relate to how the system behaves as the number of servers K increases.

- **There is a sweet-spot to reduce mean response time**: when the number of servers is small, all servers may be "blocked" by long jobs, preventing other jobs to progress. This favors an increase in the number of servers. In contrast, when the number of servers is large, a significant fraction of the servers will be idle, favoring a reduction in the number of servers. Together, these two observation imply that there is an optimal number of servers that minimizes mean response time.
- **There is a sweet-spot to reduce variance of response time**: we discovered that for all the considered setups there is an optimal number of servers that minimizes the response time variance of a typical job. Throughout this paper, the response time variance refers to the aggregate response time across all servers, but most of the results and insights apply per server as well, as we consider a homogeneous system wherein all servers have the same service capacity.
- **Reduction in number of servers favors increased response time predictability:** for all the considered scenarios except one, the number of servers that minimizes response time variance is smaller than or equal to the one that minimizes the response time mean. Therefore, for security purposes it may be worth slightly sacrificing mean response time to reduce response time variance.

The variability in service times is usually a key component that impacts the detection of attacks—reducing variance builds security. In summary, our model suggests that a system with more servers is typically more scalable, but less robust against attacks (presenting higher variability in service times).

The formal analysis of the $M/G/K$ system turns out to be a daunting task. For this reason, most of the previous works resorted to approximations and numerical analysis to get insights from the model [1], [20]. In this paper, we take a similar approach, and leave the formal proof of the conjectures implied by the insights above as subject for future work.

IV. Model parameterization: lab experiments

The parameterization of the proposed model involves the determination of the parameters presented in Table I. In this paper, we vary all the parameters to illustrate their impact on the metrics of interest, which are the mean and the standard deviation of the response times.

We run controlled experiments to assess the values of $E(X_S)$ and $E(X_L)$ under known attacks such as those produced by the Mirai botnet [21], [22]. Then, we vary λ and α to illustrate their impact on the metrics of interest.

TABLE II: Table contrasting the performance model from [1] with its security instantiation introduced in this paper

	performance model	security model
long jobs (highly demanding service)	long tasks	attacks/intrusions
system load, ρ	accounts for normal and long tasks	accounts for normal tasks and attacks
probability of anomalous service time, $1-\alpha$	probability of long service time	probability of arrival being due to an attack
standard deviation of service time	important for provisioning purposes	important for attack detection purposes
optimal number of servers	reduces mean response time	must account for standard deviation of response time

Different events may cause an increase in response time. In this paper, we focus on security threats, i.e., intrusions due to Mirai botnets. Nonetheless, the framework is general, and can encompass both endogenous and exogenous attack vectors [23]. Future work consists in extending the considered experiments to experimentally account for a wider set of vectors.

1) Testbed setup: Next, we describe the experimental setup considered in our testbed to assess the values of $E(X_S)$ and $E(X_L)$ under a typical Mirai attack. Our testbed comprised three machines, used as (i) a web-server (running the Sock Shop sample application), referred to as `sockshop`,[3] (ii) a load generator (to generate load to the web-server) [12], and (iii) a Mirai bot. The three machines were connected to a Cisco switch, through a 100 Mb/s full duplex network.

We summarize the key aspects of our experimental setup as follows:

1) Three physical machines were connected to the same Ethernet link;
2) The first machine, *Odin*, was set as a web server, working as a virtual store (`sockshop`), over the *Docker* platform;
3) The second machine, *Freya*, was set as load generator, using *Apache JMeter* with 10 threads and 10 requests per second;
4) The third machine, *Delfos*, was set as the attacker, running *Mirai* bot with the HTTP attack vector set to 256 threads, under the GET method.

2) Testbed results: We repeated each of our experiments for five runs. Each run lasted for 10 minutes when considering a setup without attacks, and 7 minutes when considering attacks.

When considering a setup without attacks, standard workload starts to be generated at time zero. The mean and standard deviation of response times are continuously updated throughout the experiment. Then, we recorded the accumulated mean and standard deviation of response times as observed at the end of the experiment, i.e., after 10 minutes.

When considering attacks, standard workload also starts to be generated at time zero. We wait for a warm-up of 3 minutes, before starting a 5 minute attack. The mean and standard deviation of response times are continuously updated throughout the experiment, and we report the accumulated measurements collected 7 minutes after the experiment started.

In our baseline lab experiments with low load and without intrusions we found that the response time mean and the standard deviation equal to 54.13ms and 39.23ms, respectively. With intrusions, those values increased to 95.20ms and 83.72ms, respectively. Recall that $E(X_L)$ (resp., $E(X_S)$) are the service times of jobs corresponding to attacks (resp., normal jobs). As in our baseline setup the normal system load was low, we assume that the response times measured in our experiments without attacks correspond to normal service times. We further assume that the service time of jobs corresponding to attacks is given by the response time of jobs under attacks, as measured in our experiments. Then, it follows that

$$E(X_S) = 54.13, \quad E(X_L) = 95.20. \tag{8}$$

All values above are measured in milliseconds. In what follows, such values are used to estimate the average and the standard deviation of the response times, $E(T)$ and $\sigma(T) = \sqrt{V(T)}$, under various scenarios of interest.

V. Findings from analytical model

Next, we assess the impact of the infrastructure setup on the performance and security of the system. In particular, we focus on the impact of the number of servers K on the mean and standard deviation of the response time. Our aim is to address the following question: what are the recommended values of K when the expected load level ρ and the fraction of load that is due to attacks, $1-\alpha$, are taken into account?[4]

A. Numerical results

Figures 4 and 5, together with Tables III and IV, respectively, present the numerical results obtained with the analytical model. Figure 4 reports results assuming $E(X_S)/E(X_L) = 0.5686$ as estimated by our experimental results (see Section IV-2). In Figure 5, we maintain $E(X_S) = 54.13$ and vary $E(X_L)$ so as to reach a target $E(X_S)/E(X_L)$.

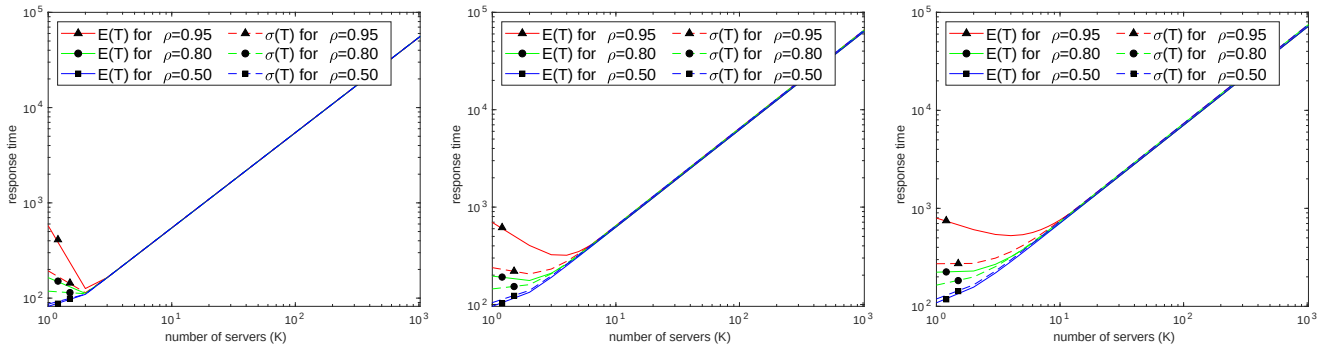
Solid (resp., dashed) lines in Figures 4 and 5 show the average (resp., standard deviation) of the response time as a function of the number of servers in the system, K. Each subplot in a given line, from left to right, corresponds to increasing fraction of jobs due to intrusions, with $1-\alpha$ varying between 0.01, 0.2 and 0.4. In Figure 5 subplots in a column correspond to increasing values of $E(X_S)/E(X_L)$.

B. Reducing number of servers favors variance reduction

The value of K that minimizes the standard deviation of the response time is typically smaller than or equal to the value of K that minimizes mean response times (see Figures 4 and 5, and Tables III and IV). This suggests that to simplify intrusion detection one must trade between security and performance. A

[3] `sockshop`: https://github.com/microservices-demo/microservices-demo

[4] All the data on which we base our analysis will be made available as an artefact on Zenodo (https://zenodo.org).



(a) $\alpha = 0.99$ (b) $\alpha = 0.80$ (c) $\alpha = 0.60$

Fig. 4: Response times as a function of system parameters. As the probability of intrusions increases from left to right ($1 - \alpha = 0.01, 0.2, 0.4$), the optimal number of servers increases (see also Table III). The ratio $E(X_S)/E(X_L) = 0.5686$ is parametrized using controlled experiments (see Section IV).

TABLE III: Optimal number of servers minimizing average and standard deviation of response time (setup of Fig. 4, reflecting controlled experiments, $E(X_S)/E(X_L) = 0.5686$).

Setup Fig. 4	$\rho = 0.5$		$\rho = 0.8$		$\rho = 0.95$	
	$K^\star_\mu$	$K^\star_\sigma$	$K^\star_\mu$	$K^\star_\sigma$	$K^\star_\mu$	$K^\star_\sigma$
4a)	1	1	2	2	2	2
4b)	1	1	2	1	4	2
4c)	1	1	1	1	4	1

notable exception consists of the setup of Fig. 5(g) with $\rho = 0.5$, for which the value of K that minimizes response time variance (resp., mean) is 2 (resp., 1), as shown in Table IV.

Smaller values of K make the system more predictable, at the expense of an increase in mean response times. As the fraction of intrusions increases, i.e., as α decreases, we observe that to minimize the standard deviation of the response time one needs to consider a smaller number of servers. In particular, for $\alpha = 0.6$ the standard deviation of response times is minimum when $K = 1$ for all the considered scenarios (see Tables III and IV).

C. Reducing number of servers is beneficial under intrusions

1) Impact of intrusion probability: Next, we consider the impact of intrusion probability, $1 - \alpha$, on response times. As $1 - \alpha$ increases, the optimal number of servers increases when considering the model parameterized by our experiments (Figure 4). That holds for large values of the ratio $E(X_S)/E(X_L)$ (i.e, when the ratio equals 0.05 or 0.5686). For small values of the ratio $E(X_S)/E(X_L)$ (i.e., when the ratio equals 0.0005 or 0.005), in contrast, the optimal number of servers decreases as $1 - \alpha$ increases (two top rows of Figure 5, corresponding to top six rows of Table IV).

2) Impact of system load: As the system load increases, the optimal number of servers consistently increases. Figures 4 and 5, and Tables III and IV, show that the values of K that minimize the mean and the standard deviation of the response times increase as the load increases for all the considered parameters. In particular, for low loads ($\rho = 0.5$), when $\alpha \in \{0.8, 0.6\}$ the optimal number of servers equals 1 or 2 under all the considered scenarios (two last columns of Figures 4 and 5).

D. Model validation through simulations

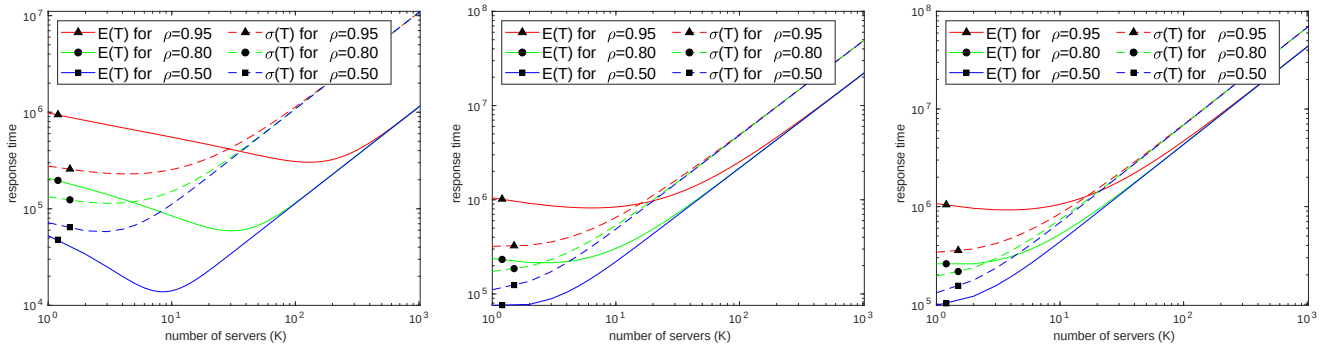
To validate the model we simulated all the considered scenarios and contrasted simulation results against our approximate analytic model. Our simulations are executed using an efficient simulator which leverages an extension of recursive Lindley equations [24], allowing us to produce numerical results more efficiently than traditional event driven simulations. Figure 6 and Table IV report a sample of our simulation results (see [18] for a more comprehensive report).

Figure 6 shows that the proposed model approximations are more accurate for small values of ρ. When $\rho = 0.95$, model approximations still have predictive power to allow approximations for the optimal values of K that minimize $E(T)$ and $V(T)$, but they are less useful to predict the values of the metrics of interest. Note that scenarios wherein ρ is large are arguably of less relevance, as system administrators will typically provision the system to avoid heavy traffic.

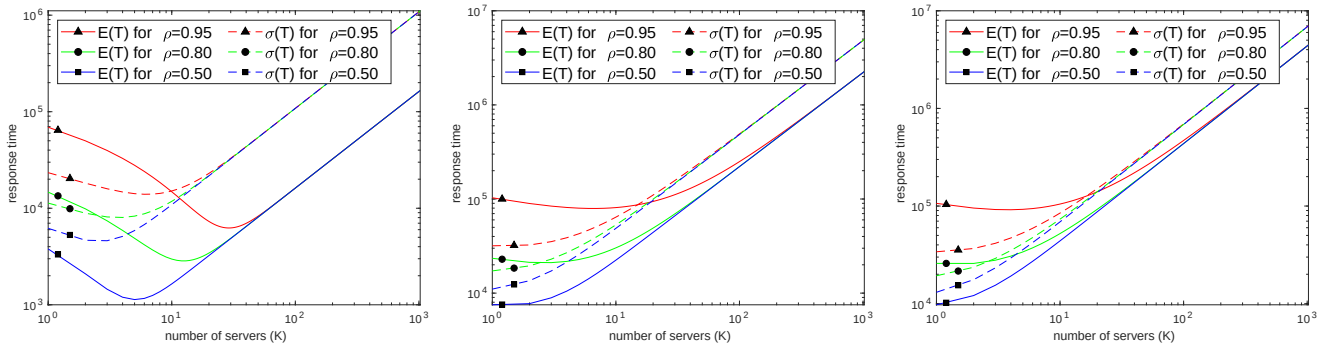
VI. Discussion and Implications

Next, we discuss some of the practical implications of our work. In this work we proposed first steps towards models and measurements to capture the tradeoff between scalability and efficient anomaly detection. The implications of that tradeoff to architecture design include the tuning of IDS systems and the setup of cloud computing infrastructures.

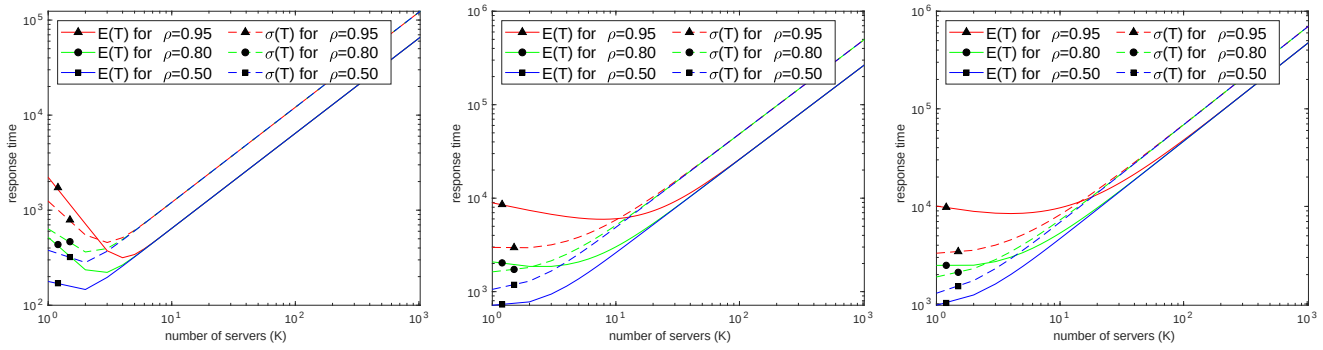
A more sensitive anomaly detection system may filter more malicious jobs, at the expense of also filtering workload from real users due to false positives. The same holds for an admission control which, by filtering more jobs, may increase the performance of the system, at the expense of indirectly causing the blocking of real users, which translates into a denial of service, ultimately achieving the goals of the attacker.



(a) $\alpha = 0.99, E(X_S)/E(X_L) = 0.0005$ (b) $\alpha = 0.80, E(X_S)/E(X_L) = 0.0005$ (c) $\alpha = 0.60, E(X_S)/E(X_L) = 0.0005$



(d) $\alpha = 0.99, E(X_S)/E(X_L) = 0.005$ (e) $\alpha = 0.80, E(X_S)/E(X_L) = 0.005$ (f) $\alpha = 0.60, E(X_S)/E(X_L) = 0.005$



(g) $\alpha = 0.99, E(X_S)/E(X_L) = 0.05$ (h) $\alpha = 0.80, E(X_S)/E(X_L) = 0.05$ (i) $\alpha = 0.60, E(X_S)/E(X_L) = 0.05$

Fig. 5: Response times as a function of system parameters. The number of servers that reduces the standard deviation of response times is typically smaller than or equal to the number of servers that reduces mean response times, indicating that to simplify intrusion detection one must trade between security and performance.

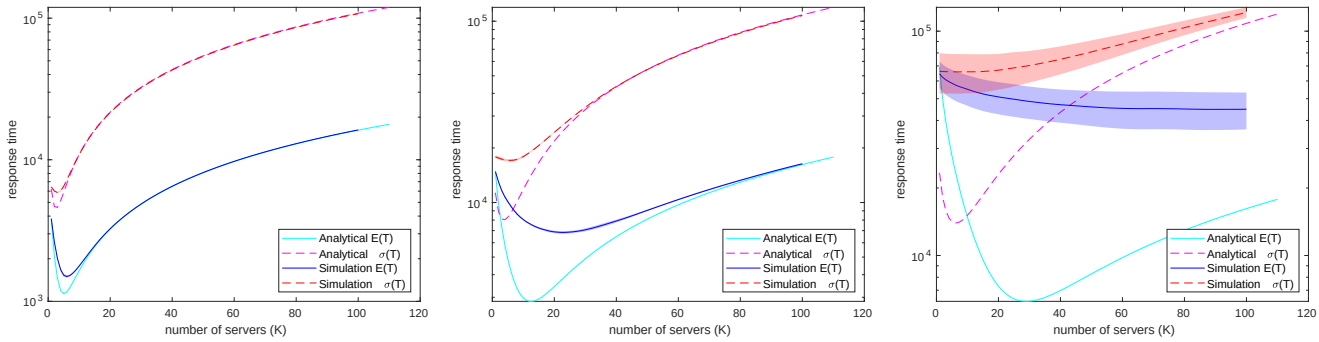
From the infrastructure setup perspective, increasing the number of servers builds scalability, but may also increase the standard deviation of response times. We analytically observed that trend in all the considered scenarios. We experimentally investigated the impact of the workload and of the infrastructure on domain metrics, and discovered that those metrics can be helpful (and in some cases sufficient) to detect attacks and to capture tradeoffs between scalability and anomaly detection.

VII. Related Work

There is a vast literature on analytical and experimental aspects related to security and performance [9], [25], [26]. Nonetheless, to the best of our knowledge this work is the first to analytically study the interplay between host-based anomaly detection leveraging customer affecting metrics and system scalability in multi-server systems.

A. Multi-server queues and optimal number of servers

The use of the $M/G/K$ queue to model cloud multi-server systems has been considered in [1], [27]. In this paper, we apply the model introduced in [1] to the performance analysis of denial of service attacks in multi-server systems. Security implications of [1] have been briefly pointed out by [28]. In this work, we take a step further, studying how the mean and



(a) $\rho = 0.5$ (b) $\rho = 0.8$ (c) $\rho = 0.95$

Fig. 6: Model validation: response times as a function of system load ($\rho = 0.5, 0.8, 0.95$). Analytical and simulation results corresponding to the setup in Figure 5a). As ρ increases, the approximations become less accurate. When $\rho = 0.95$, model approximations still have predictive power to allow approximations for the optimal values of K that minimize $E(T)$ and $V(T)$.

TABLE IV: Optimal number of servers minimizing average and standard deviation of response time ($K^\star_\mu$ and $K^\star_\sigma$, resp.). Confidence intervals and additional statistics available at [18].

Fig. 5		analytical (simulation)			
	ρ	$K^\star_\mu$	$K^\star_\sigma$	$\mu^\star$	$\sigma^\star$
a	0.95	12(100)	4(7)	(586306.11)	(725673.14)
a	0.80	12(39)	3(7)	(97052.11)	(259498.89)
a	0.50	8(9)	3(3)	(15331.20)	(72473.12)
b	0.95	6(2)	1(1)	(1435379.09)	(1596452.87)
b	0.80	3(1)	1(1)	(234981.57)	(252471.25)
b	0.50	1(1)	1(1)	(76085.60)	(93051.78)
c	0.95	4(1)	1(1)	(1047949.40)	(1014490.52)
c	0.80	2(1)	1(1)	(259224.59)	(258537.61)
c	0.50	1(1)	1(1)	(96914.92)	(97411.19)
d	0.95	12(87)	6(9)	(34744.17)	(51828.85)
d	0.80	12(22)	4(4)	(6583.95)	(15603.64)
d	0.50	5(5)	3(3)	(1426.94)	(5637.30)
e	0.95	7(2)	1(1)	(94903.73)	(91224.34)
e	0.80	3(1)	1(1)	(23225.27)	(24055.67)
e	0.50	1(1)	1(1)	(7528.79)	(9245.52)
f	0.95	4(1)	1(1)	(113102.15)	(113640.00)
f	0.80	2(1)	1(1)	(25947.68)	(25473.96)
f	0.50	1(1)	1(1)	(9748.61)	(9810.14)
g	0.95	4(3)	3(3)	(2227.73)	(2456.33)
g	0.80	3(2)	2(2)	(513.02)	(672.40)
g	0.50	2(1)	2(2)	(177.61)	(278.75)
h	0.95	8(1)	2(1)	(8173.49)	(7377.48)
h	0.80	2(1)	1(1)	(2090.42)	(2197.18)
h	0.50	1(1)	1(1)	(705.02)	(822.09)
i	0.95	4(1)	1(1)	(10162.91)	(9746.30)
i	0.80	1(1)	1(1)	(2519.10)	(2458.63)
i	0.50	1(1)	1(1)	(975.46)	(937.73)

standard deviation of response times may impact decisions related to the setup of service infrastructures.

The problem of determining the optimal number of servers in a computer system is one of the most classical problems in queuing theory and autonomic computing [29], [30]. The optimal number of servers in $M/G/K$ queues and its variants has also received some attention in the past [20], [31], [32]. However, none of those previous works accounted for the response time predictability, which is intrinsically related to system security, when tuning the number of servers. In this paper, in contrast, we identify a fundamental tradeoff between performance and security when accounting for the mean and variance of response times.

In the monitoring literature, mean response time is typically considered a limited metric as response times are heavy-tailed [33], [34]. Therefore, researchers usually suggest the use of quartile-based metrics, e.g., for the specification of service-level objectives or the configuration of performance anomaly detection systems. Nonetheless, most of the previous work on analytical models focused primarily on mean response times [3], [4]. In this paper, we contribute to bridging that gap between theory and practice, by further investigating the practical implications of the response time variance as predicted by the analyical model.

B. Anomaly detection

In [35] a survey of anomaly detection techniques in several application domains was presented. The survey paper identifies some of the most important application domains for anomaly detection application as intrusion detection, fraud detection, fault-detection, and medical diagnosis. The authors have enumerated the most important challenges in anomaly detection research as follows: (i) it is very difficult to comprehensively define normal behavior, (ii) malicious attackers may adapt their behavior to fit the domain definition of "normal behavior", (iii) anomaly definition varies per domain, (iv) data availability for anomaly training is not easy to obtain, (v) training data is usually noisy. As a consequence of these challenges, researchers usually develop heuristics for anomaly detection that take advantage of specific characteristics of the application domain.

In this paper, we have attempted to address these concerns by: (i) using an analytic model to capture normal behavior in terms of mean and variance of response time, (ii) focus on denial of service attacks, which have impact on mean and variance of the response time, (iii) focus on a given

performance engineering domain, (iv) as our anomaly detection algorithm uses mean and variance of response time, our approach does not require extensive training data, (v) use a controlled environment to avoid training data noise.

In [36], [37] and [38] the authors propose threshold-based and more general rule-based models for intrusion detection systems. The derivation of optimal thresholds and meaningful rules is typically obtained through classical statistical methods or machine learning. In this paper, in contrast, we take advantage of an analytic model and an existing performance testing infrastructure to study some fundamental tradeoffs between anomaly detection and scalability. Our approach relies on experiments that are run in a controlled environment to derive the parameters required to calibrate the performance model and the anomaly detection algorithm.

C. Security-scalability tradeoff

In [10], [39] the authors present a framework for the simultaneous scalability and security confidentiality assessment for data intensive web applications. The authors presented strategies for database view invalidation to help select which data shall be encrypted without impacting scalability when using a database service provider. They have proposed a new scalability aware security design using: (1) mandatory encryption of sensitive information, and, (2) restricting data encryption to encrypt only data that is not scalability affecting. Our work is related to [10], as we also propose to use a security aware scalability approach. However, in this paper we analyze how to account for tradeoffs between anomaly detection and scalability when computing the optimal number of servers in multi-server central-queueing systems.

In [40] the authors also discuss the security-performance tradeoff: increasing security typically degrades performance, as screening users may slow down the site. As pointed above, this may ultimately translate into a denial of service, achieving the goals of the attacker. In this paper, we point that system administrators may control the server infrastructure (e.g., by adjusting the number of servers accounting for the standard deviation of response times) in order to circumvent the challenge of inadvertently slowing down users in face of potential attacks.

D. IDS tools

The literature on IDS tools is vast [5], [6], [41], [42], [43], [42]. A comprehensive survey of IDS tools is presented in [5], where three properties of IDS tools are used to classify those systems: (i) monitored platform (host based, network based, or hybrid), (ii) attack detection method (misuse based, anomaly based, hybrid), and, (iii) deployment architecture (non-distributed and distributed). In the anomaly based IDSs, a baseline profile of normal operations is developed and deviations from the baseline profile are identified as intrusions using performance signatures.

Avritzer et al. [7] proposed an architecture for intrusion detection systems using off-the-shelf IDSs complemented by performance signatures. The authors have shown that the performance signature of well-behaved systems and of several types of security attacks could be identified in terms of certain performance metrics, such as CPU and memory percentage, number of active threads, etc. Specifically, the following security attacks were evaluated positively for performance signature detection, when compared against off-the-shelf IDS detection: man in the middle, denial of service, buffer overflow, stack overflow, and SQL injection.

The performance of signature-based intrusion detection systems rely on intrusion detection algorithms that account for workload variability to avoid a high rate of false positive alerts. An example of such workload-sensitive algorithms is the bucket algorithm [14]. Anomaly detection using the Bucket algorithm inspects the last sample of response time and compare it against calibrated mean and standard deviation of response time. Then, it relies on the central limit theorem [44] to assess if the last samples of response time comply with normal behavior or represent an anomaly. We envision that the models, measurements and insights presented in this paper are instrumental assist in setting the baselines to those algorithms.

VIII. Conclusions

Anomaly detection has been widely recognized as an important production tool in several application domains, such as, computer networks, medical, banking, and failure diagnosis in mission-critical domains [35]. In this paper, we have introduced an anomaly detection framework that is supported by standard performance engineering process steps, such as workload modeling, performance modeling, performance testing, and performance monitoring. Our focus is on one of the key elements of the proposed framework: an analytic model to estimate the average and standard deviation of response time of a central server system subjected to security attacks. We have used the performance testing environment to derive parameters for the calibration of the proposed analytic model. Our original experiments are encouraging, showing that the customer-affecting metric is sensitive to the security intrusions evaluated.

We have found that the number of servers that minimizes response time variance is typically smaller than or equal to the one that minimizes the response time mean. Therefore, for security purposes it may be worth slightly sacrificing mean response time to reduce its variance. The variability in service times is usually a key component that impacts the performance of the anomaly detection algorithm.

As topics for future research, we envision the application of the framework introduced in this paper to large mission-critical systems. Specifically, additional experiments are required to evaluate how to integrate admission control and load balancing [2], with the cloud computing infrastructure configuration approach introduced in this paper to ensure mean response time predictability. In addition, we would like to study how to best tune the anomaly detection algorithm [44] to (i) ensure that the central-limit theorem statistical hypothesis applies to the monitored data, and to (ii) provide a good statistical characterization of normal host-based usage to reduce the number

of false positives triggered by the anomaly detection algorithm. We also propose to compare the performance of anomaly detection algorithms based on different approaches, such as bucket algorithms [44] and machine learning algorithms [15].

REFERENCES

[1] K. Psounis, P. Molinero-Fernández, B. Prabhakar, and F. Papadopoulos, "Systems with multiple servers under heavy-tailed workloads," *Performance Evaluation*, vol. 62, no. 1-4, pp. 456–474, 2005.

[2] T. Nylander, M. T. Andrén, K.-E. Årzén, and M. Maggio, "Cloud application predictability through integrated load-balancing and service time control," in *IEEE International Conference on Autonomic Computing (ICAC)*, 2018.

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

[4] D. A. Menasce and V. A. Almeida, *Capacity Planning for Web Services: metrics, models, and methods*. Prentice Hall, 2002.

[5] A. Milenkoski, M. Vieira, S. Kounev, A. Avritzer, and B. D. Payne, "Evaluating computer intrusion detection systems: A survey of common practices," *ACM Computing Surveys (CSUR)*, vol. 48, no. 1, p. 12, 2015.

[6] M. Grottke, A. Avritzer, D. S. Menasché, L. P. de Aguiar, and E. Altman, "On the efficiency of sampling and countermeasures to critical-infrastructure-targeted malware campaigns," *SIGMETRICS Performance Evaluation Review*, vol. 43, no. 4, pp. 33–42, 2016.

[7] A. Avritzer, R. Tanikella, K. James, R. G. Cole, and E. Weyuker, "Monitoring for security intrusion using performance signatures," in *Proceedings of the first joint WOSP/SIPEW International Conference on Performance Engineering*. ACM, 2010, pp. 93–104.

[8] K. Wolter and P. Reinecke, "Performance and security tradeoff," in *Proceedings of the Formal Methods for Quantitative Aspects of Programming Languages, and 10th International Conference on School on Formal Methods for the Design of Computer, Communication and Software Systems*, ser. SFM'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 135–167.

[9] F. Alomari and D. A. Menasce, "An autonomic framework for integrating security and quality of service support in databases," in *Software Security and Reliability (SERE), 2012 IEEE Sixth International Conference on*. IEEE, 2012, pp. 51–60.

[10] A. Manjhi, A. Ailamaki, B. M. Maggs, T. C. Mowry, C. Olston, and A. Tomasic, "Simultaneous scalability and security for data-intensive web applications," in *International Conference on Management of Data (SIGMOD)*. ACM, 2006, pp. 241–252.

[11] C. Heger, A. van Hoorn, M. Mann, and D. Okanovic, "Application performance management: State of the art and challenges for the future," in *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering (ICPE 2017)*, 2017, pp. 429–432.

[12] A. Avritzer, V. Ferme, A. Janes, B. Russo, H. Schulz, and A. van Hoorn, "A quantitative approach for the assessment of microservice architecture deployment alternatives using automated performance testing," in *Proceedings of the 12th European Conference on Software Architecture (ECSA 2018)*, ser. LNCS. Springer, 2018, pp. 159–174.

[13] Z. M. Jiang, A. E. Hassan, G. Hamann, and P. Flora, "Automatic identification of load testing problems," in *International Conference on Software Maintenance (ICSM)*. IEEE, 2008, pp. 307–316.

[14] A. Avritzer, A. Bondi, and E. J. Weyuker, "Ensuring stable performance for systems that degrade," in *International Workshop on Software and Performance (WOSP)*. ACM, 2005, pp. 43–51.

[15] J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan, "Fast portscan detection using sequential hypothesis testing," in *Proceedings of IEEE Symposium on Security and Privacy*. IEEE, 2004, pp. 211–225.

[16] V. Gupta, M. Harchol-Balter, J. Dai, and B. Zwart, "On the inapproximability of m/g/k: why two moments of job size distribution are not enough," *Queueing Systems*, vol. 64, no. 1, pp. 5–48, 2010.

[17] I. Grosof, Z. Scully, and M. Harchol-Balter, "SRPT for multiserver systems," *arXiv preprint arXiv:1805.07686*, 2018.

[18] A. Avritzer, V. Rufino, D. Menasché, B. Russo, A. Janes, V. Ferme, A. van Hoorn, and H. Schulz, "Companion technical report and scripts," 2019, https://github.com/queupe/mgk_simulation/.

[19] S. L. Brumelle, "Some inequalities for parallel-server queues," *Operations Research*, vol. 19, no. 2, pp. 402–413, 1971.

[20] A. Wierman, T. Osogami, M. Harchol-Balter, and A. Scheller-Wolf, "How many servers are best in a dual-priority m/ph/k system?" *Performance Evaluation*, vol. 63, no. 12, pp. 1253–1272, 2006.

[21] G. Kambourakis, C. Kolias, and A. Stavrou, "The Mirai botnet and the IoT zombie armies," in *Proceedings of the Military Communications Conference (MILCOM 2017)*. IEEE, 2017, pp. 267–272.

[22] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis *et al.*, "Understanding the mirai botnet," in *USENIX Security Symposium*, 2017, pp. 1092–1110.

[23] A. Laszka, M. Felegyhazi, and L. Buttyán, "A survey of interdependent security games," *CrySyS*, vol. 2, 2012.

[24] W. Kin and V. Chan, "Generalized Lindley-type recursive representations for multiserver tandem queues with blocking," *Transactions on Modeling and Computer Simulation*, vol. 20, no. 4, p. 21, 2010.

[25] X. Zhao, Q. Lin, J. Chen, X. Wang, J. Yu, and Z. Ming, "Optimizing security and quality of service in a real-time database system using multi-objective genetic algorithm," *Expert Systems with Applications*, vol. 64, pp. 11–23, 2016.

[26] F. Alomari and D. A. Menasce, "Efficient response time approximations for multiclass fork and join queues in open and closed queuing networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 6, pp. 1437–1446, 2014.

[27] H. Khazaei, J. Misic, and V. B. Misic, "Performance analysis of cloud computing centers using m/g/m/m+ r queuing systems," *IEEE Transactions on parallel and distributed systems*, vol. 23, no. 5, pp. 936–943, 2012.

[28] M. Rohr, "Workload-sensitive timing behavior analysis for fault localization in software systems," Dissertation (PhD thesis), Faculty of Engineering, Kiel University, Kiel, Germany, Jan. 2015.

[29] S. Stidham Jr, "On the optimality of single-server queuing systems," *Operations Research*, vol. 18, no. 4, pp. 708–732, 1970.

[30] U. Sharma, P. Shenoy, and D. F. Towsley, "Provisioning multi-tier cloud applications using statistical bounds on sojourn time," in *International conference on Autonomic computing (ICAC)*. ACM, 2012, pp. 43–52.

[31] S. Qin, "A new optimal cost model of queue systems with heavy-tailed distribution," in *Information Science and Engineering (ICISE), 2010 2nd International Conference on*. IEEE, 2010, pp. 2530–2533.

[32] A. Scheller-Wolf, "Necessary and sufficient conditions for delay moments in fifo multiserver queues with an application comparing s slow servers with one fast one," *Operations Research*, vol. 51, no. 5, pp. 748–758, 2003.

[33] A. Mielke, "Elements for response-time statistics in ERP transaction systems," *Perform. Eval.*, vol. 63, no. 7, pp. 635–653, Jul. 2006.

[34] M. Rohr, A. van Hoorn, W. Hasselbring, M. Lübcke, and S. Alekseev, "Workload-intensity-sensitive timing behavior analysis for distributed multi-user software systems," in *Joint International Conference on Performance Engineering*. ACM, 2010, pp. 87–92.

[35] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, pp. 15:1–15:58, 2009.

[36] V. K. Gurbani, D. Kushnir, V. B. Mendiratta, C. Phadke, E. Falk, and R. State, "Detecting and predicting outages in mobile networks with log data," in *Proceedings of the IEEE International Conference on Communications (ICC 2017)*, 2017, pp. 1–7.

[37] V. Kumar, "Parallel and distributed computing for cybersecurity," *IEEE Distributed Systems Online*, vol. 6, 2005.

[38] D. E. Denning, "An intrusion-detection model," *IEEE Transactions on Software Engineering*, vol. SE-13, no. 2, pp. 222–232, Feb 1987.

[39] A. Manjhi, "Increasing the scalability of data-intensive web applications," 2006, http://www.cs.cmu.edu/~manjhi/thesis/proposal.pdf.

[40] D. Gillman, Y. Lin, B. Maggs, and R. K. Sitaraman, "Protecting websites from attack with secure delivery networks," *Computer*, vol. 48, no. 4, pp. 26–34, 2015.

[41] R. Mitchell and I.-R. Chen, "A survey of intrusion detection techniques for cyber-physical systems," *ACM Comput. Surv.*, vol. 46, no. 4, pp. 55:1–55:29, Mar. 2014.

[42] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in internet of things," *Journal of Network and Computer Applications*, vol. 84, pp. 25 – 37, 2017.

[43] M. Felemban, Y. Javed, J. Kobes, T. Qadah, A. Ghafoor, and W. Aref, "Design and evaluation of a data partitioning-based intrusion management architecture for database systems," 2018. [Online]. Available: https://arxiv.org/pdf/1810.02061.pdf

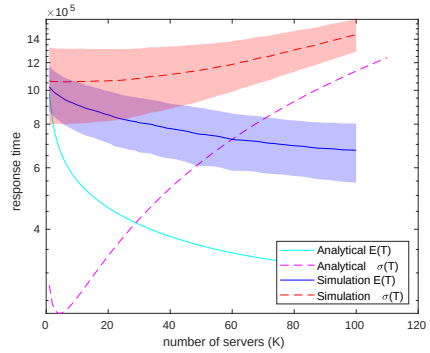
[44] A. Avritzer, R. G. Cole, and E. J. Weyuker, "Using performance signatures and software rejuvenation for worm mitigation in tactical MANETs," in *International Workshop on Software and Performance Engineering*, 2007, pp. 172–180.

APPENDIX A
ADDITIONAL NUMERICAL RESULTS

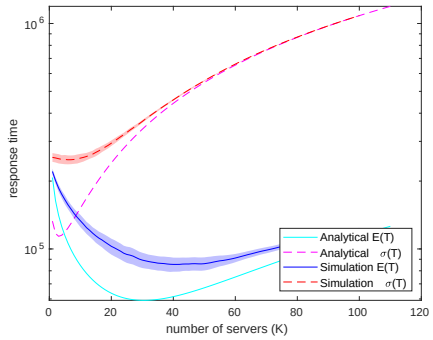
Next, we report additional numerical results obtained through our analytical model approximations and simulations.

TABLE V: Optimal number of servers minimizing average and standard deviation of response time ($K_\mu^\star$ and $K_\sigma^\star$, resp.), the value of the average and standard deviation of response time , and the average and standard deviation of response time for one server.

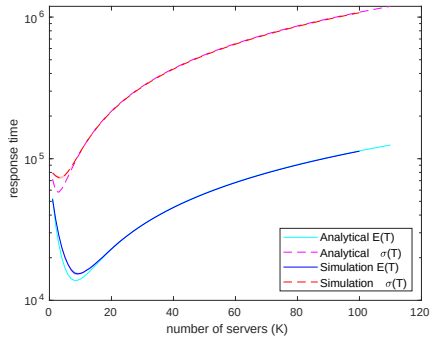
E_S/E_L	α	ρ	analytical (simulation)					
			$K_\mu^\star$	$K_\sigma^\star$	$\mu^\star$	$\sigma^\star$	$\mu_{K=1}$	$\sigma_{K=1}$
0.0005	0.99	0.95	110(100)	4(6)	305779.58(673511.58)	229709.60(1057153.18)	(981122.33)	(276771.84)
0.0005	0.99	0.80	30(41)	3(6)	59155.35(85381.27)	113757.70(248643.70)	(207449.06)	(132850.39)
0.0005	0.99	0.50	8(9)	3(3)	13832.47(15383.40)	57995.10(73628.48)	(52714.41)	(71838.26)
0.0005	0.80	0.95	6(2)	1(1)	819009.29(1076053.40)	320591.69(1171635.82)	(1048113.50)	(320591.69)
0.0005	0.80	0.80	3(1)	1(1)	214297.84(235356.55)	173298.43(251482.63)	(237783.34)	(173298.43)
0.0005	0.80	0.50	1(1)	1(1)	75717.31(75562.99)	110856.90(93023.28)	(75717.31)	(110856.90)
0.0005	0.60	0.95	4(1)	1(1)	924227.47(1051415.46)	340815.90(1033144.64)	(1071036.09)	(340815.90)
0.0005	0.60	0.80	2(1)	1(1)	259296.47(261490.71)	193430.68(257243.25)	(259694.29)	(193430.68)
0.0005	0.60	0.50	1(1)	1(1)	97425.93(97330.07)	130950.16(98112.69)	(97425.93)	(130950.16)
0.0050	0.99	0.95	29(90)	6(7)	6246.33(44824.63)	13960.01(65835.37)	(69126.09)	(23366.56)
0.0050	0.99	0.80	12(23)	4(6)	2850.30(6838.74)	8029.51(17074.55)	(14680.64)	(11307.91)
0.0050	0.99	0.50	5(6)	3(3)	1132.58(1496.67)	4608.90(5863.73)	(3791.55)	(6195.92)
0.0050	0.80	0.95	7(2)	1(1)	79372.55(99450.78)	31818.19(100033.26)	(103048.98)	(31818.19)
0.0050	0.80	0.80	3(1)	1(1)	21086.50(23491.16)	17219.40(24731.14)	(23438.08)	(17219.40)
0.0050	0.80	0.50	1(1)	1(1)	7515.90(7505.94)	11030.59(9222.81)	(7515.90)	(11030.59)
0.0050	0.60	0.95	4(1)	1(1)	91616.80(106977.06)	33990.33(107257.79)	(106448.10)	(33990.33)
0.0050	0.60	0.80	2(1)	1(1)	25826.31(25898.44)	19301.27(25242.58)	(25854.50)	(19301.27)
0.0050	0.60	0.50	1(1)	1(1)	9735.78(9741.73)	13074.18(9760.89)	(9735.78)	(13074.18)
0.0500	0.99	0.95	4(3)	3(3)	316.11(2234.81)	457.04(2472.44)	(2220.74)	(1244.74)
0.0500	0.99	0.80	3(2)	2(3)	221.60(503.28)	363.42(662.72)	(518.38)	(636.56)
0.0500	0.99	0.50	2(1)	2(2)	146.39(176.64)	281.72(277.23)	(177.91)	(378.74)
0.0500	0.80	0.95	8(1)	2(1)	5948.63(8698.78)	2950.16(8540.61)	(8916.11)	(2974.29)
0.0500	0.80	0.80	2(1)	1(1)	1861.65(2071.57)	1628.01(2175.33)	(2082.20)	(1628.01)
0.0500	0.80	0.50	1(1)	1(1)	715.42(712.31)	1057.29(834.95)	(715.42)	(1057.29)
0.0500	0.60	0.95	4(1)	1(1)	8445.50(10174.89)	3313.95(10272.73)	(10068.56)	(3313.95)
0.0500	0.60	0.80	1(1)	1(1)	2487.21(2478.14)	1891.78(2403.31)	(2487.21)	(1891.78)
0.0500	0.60	0.50	1(1)	1(1)	970.94(970.90)	1288.88(934.07)	(970.94)	(1288.88)



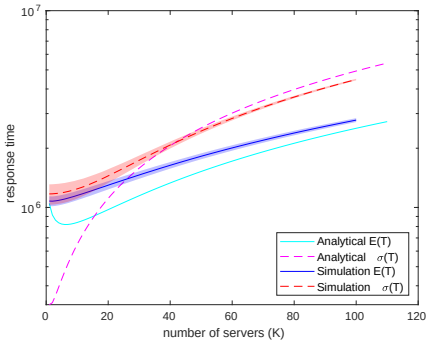
(a) $\alpha = 0.99$, $\rho = 0.95$ $E(X_S)/E(X_L) = 0.0005$,



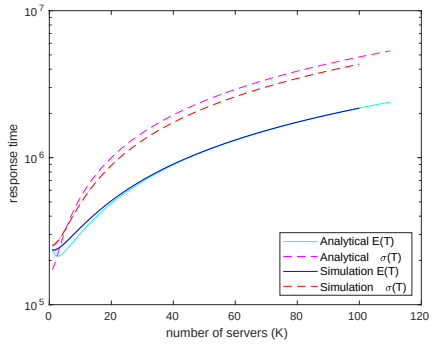
(b) $\alpha = 0.99$, $\rho = 0.80$ $E(X_S)/E(X_L) = 0.0005$,



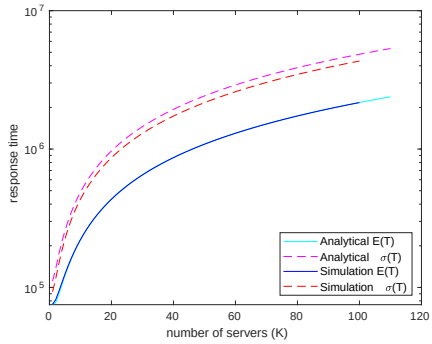
(c) $\alpha = 0.99$, $\rho = 0.50$ $E(X_S)/E(X_L) = 0.0005$,



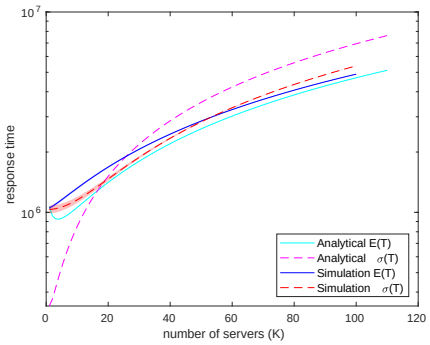
(d) $\alpha = 0.80$, $\rho = 0.95$ $E(X_S)/E(X_L) = 0.0005$,



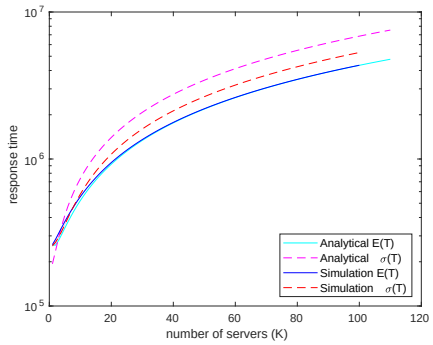
(e) $\alpha = 0.80$, $\rho = 0.80$ $E(X_S)/E(X_L) = 0.0005$,



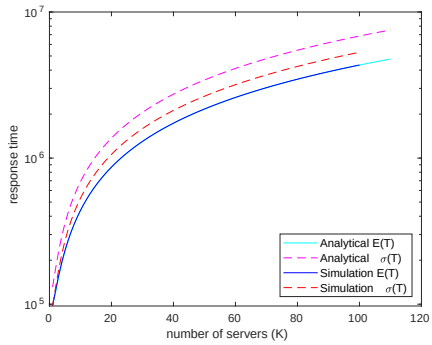
(f) $\alpha = 0.80$, $\rho = 0.50$ $E(X_S)/E(X_L) = 0.0005$,



(g) $\alpha = 0.60$, $\rho = 0.95$ $E(X_S)/E(X_L) = 0.0005$,

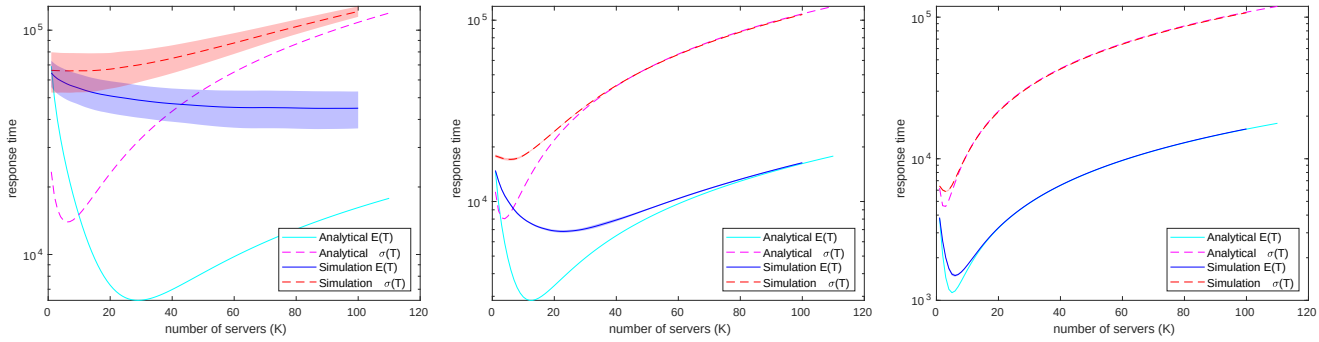


(h) $\alpha = 0.60$, $\rho = 0.80$ $E(X_S)/E(X_L) = 0.0005$,



(i) $\alpha = 0.60$, $\rho = 0.50$ $E(X_S)/E(X_L) = 0.0005$,

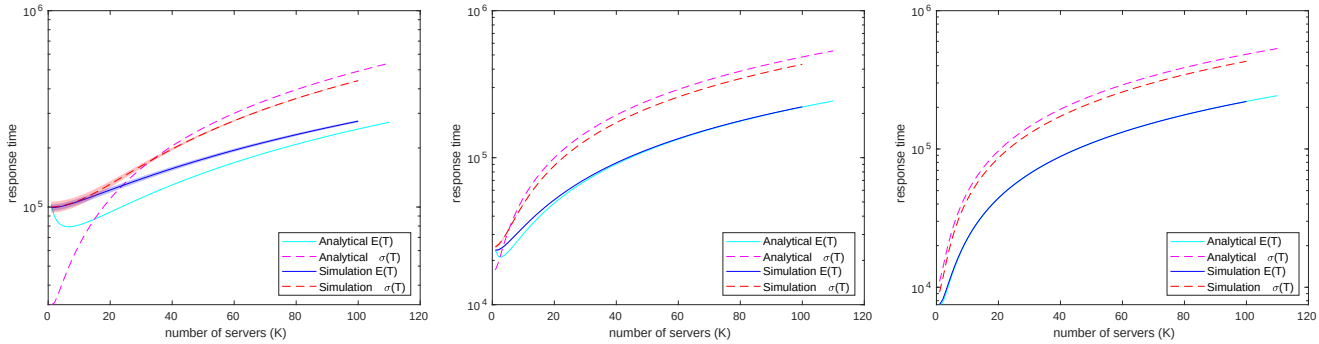
Fig. 7: Response times as a function of system parameters, for analytical and simulation analises.



(a) $\alpha = 0.99$, $\rho = 0.95$ $E(X_S)/E(X_L) = 0.0050$,

(b) $\alpha = 0.99$, $\rho = 0.80$ $E(X_S)/E(X_L) = 0.0050$,

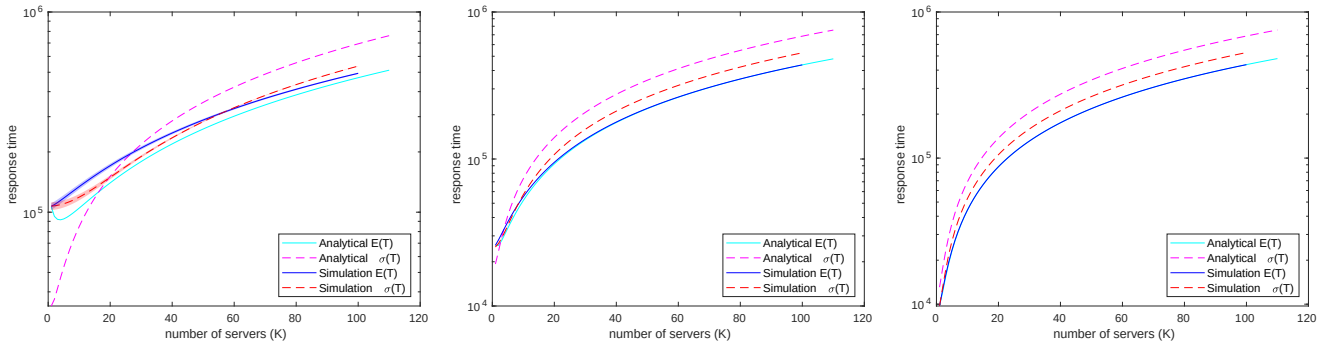
(c) $\alpha = 0.99$, $\rho = 0.50$ $E(X_S)/E(X_L) = 0.0050$,



(d) $\alpha = 0.80$, $\rho = 0.95$ $E(X_S)/E(X_L) = 0.0050$,

(e) $\alpha = 0.80$, $\rho = 0.80$ $E(X_S)/E(X_L) = 0.0050$,

(f) $\alpha = 0.80$, $\rho = 0.50$ $E(X_S)/E(X_L) = 0.0050$,

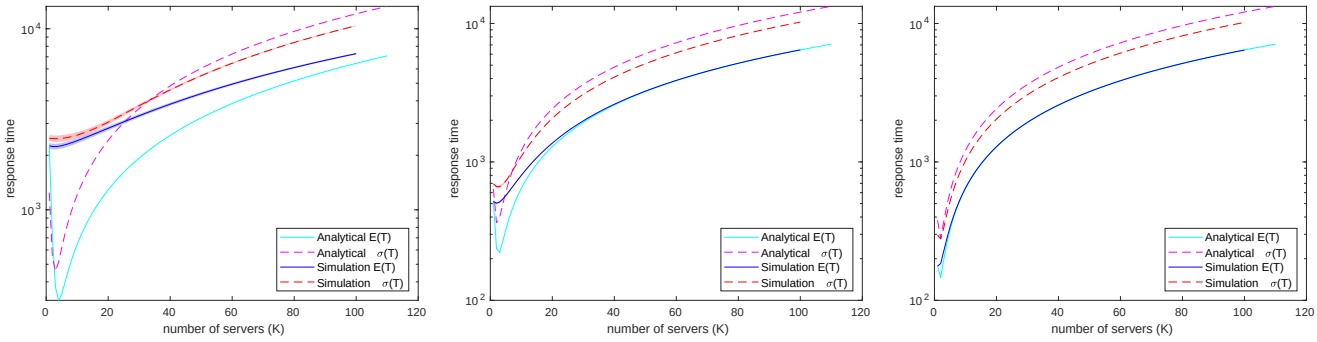


(g) $\alpha = 0.60$, $\rho = 0.95$ $E(X_S)/E(X_L) = 0.0050$,

(h) $\alpha = 0.60$, $\rho = 0.80$ $E(X_S)/E(X_L) = 0.0050$,

(i) $\alpha = 0.60$, $\rho = 0.50$ $E(X_S)/E(X_L) = 0.0050$,

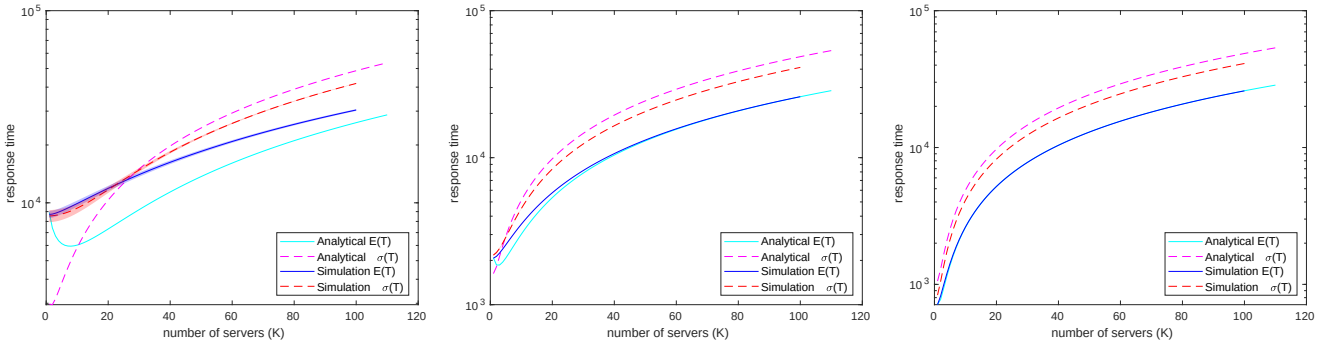
Fig. 8: Response times as a function of system parameters, for analytical and simulation analises.



(a) $\alpha = 0.99$, $\rho = 0.95$ $E(X_S)/E(X_L) = 0.0500$,

(b) $\alpha = 0.99$, $\rho = 0.80$ $E(X_S)/E(X_L) = 0.0500$,

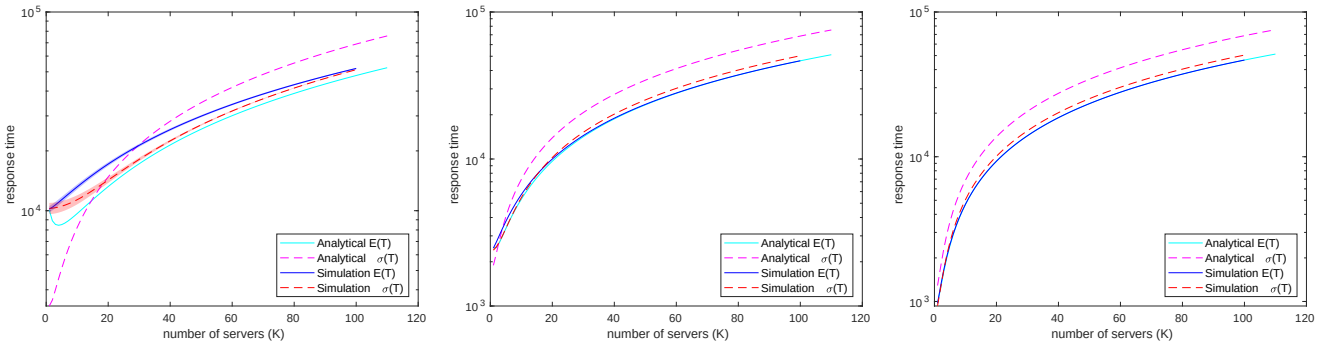
(c) $\alpha = 0.99$, $\rho = 0.50$ $E(X_S)/E(X_L) = 0.0500$,



(d) $\alpha = 0.80$, $\rho = 0.95$ $E(X_S)/E(X_L) = 0.0500$,

(e) $\alpha = 0.80$, $\rho = 0.80$ $E(X_S)/E(X_L) = 0.0500$,

(f) $\alpha = 0.80$, $\rho = 0.50$ $E(X_S)/E(X_L) = 0.0500$,



(g) $\alpha = 0.60$, $\rho = 0.95$ $E(X_S)/E(X_L) = 0.0500$,

(h) $\alpha = 0.60$, $\rho = 0.80$ $E(X_S)/E(X_L) = 0.0500$,

(i) $\alpha = 0.60$, $\rho = 0.50$ $E(X_S)/E(X_L) = 0.0500$,

Fig. 9: Response times as a function of system parameters, for analytical and simulation analises.