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

VILC RUFINO, Federal University of Rio de Janeiro (UFRJ) and Brazilian Navy, Brazil
MATEUS NOGUEIRA, DANIEL MENASCHÉ, and CABRAL LIMA, UFRJ, Brazil
ALBERTO AVRITZER, Esulab Solutions, USA
BARBARA RUSSO and ANDREA JANES, Free University of Bozen-Bolzano, Italy
VINCENZO FERME, Universita della Svizzera Italiana, Italy
ANDRÉ VAN HOORN, University of Stuttgart, Germany
HENNING SCHULZ, Novatec Consulting GmbH, Germany

Anomaly detection systems aim to detect and report attacks or unexpected behavior in networked systems. Previous work has shown that anomalies have an impact on system performance, and that performance signatures can be effectively used for implementing an IDS. In this paper, we present an analytical and an experimental study on the trade-off between anomaly 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 “long” jobs represent large resource consuming transactions, e.g., generated by DDoS attacks; the model was parametrized using results obtained from controlled experiments. For performance purposes, mean response time is the key metric to be minimized, whereas for security purposes, response time variance and classification accuracy must be taken into account. 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 and maximizes classification accuracy is typically smaller than or equal to the one that minimizes mean response time. Therefore, for security purposes, it may be worth slightly sacrificing performance to increase classification accuracy.

CCS Concepts: • Security and privacy → Intrusion detection systems.

Additional Key Words and Phrases: Cloud Computer, Anomaly Detection, Performance

1 INTRODUCTION

The multi-server paradigm with a central queueing system, e.g., encompassing a load balancing front-end towards a cloud computing environment, has been used as the state of the practice for scalable server-side computation for the last several years [36, 37, 47]. In spite of the widespread adoption of the multi-server paradigm, the setup of such systems still poses a number of challenges related to the shaping of the workload and the configuration of the servers. Security threats against multi-server systems, for instance, pose a pressing need for a better understanding of how the configuration of multi-server systems impacts their performance, security and predictability (Figure 1).

In this paper, we propose a framework for the analysis of the fundamental tradeoffs between performance, security and predictability in central-queue multi-server systems [19, 32]. In particular, we are interested in analyzing the
effectiveness of anomaly detection approaches based on performance signatures, as for example, response time. Such approaches rely on the response time being predictable.

Figure 1 shows an example of how a multi-server system, comprising a central queueing system, a load balancer and multiple servers, is integrated into an anomaly detection framework. It shows at the front-end the regular workload and attack streams being filtered by a response time based anomaly detection system. In addition, it shows at the back-end the analysis of mean response time and variance used to support response time predictability.

Performance of cloud services is typically captured through its mean response time. Anomaly detection based on performance may rely on tracking mean and variance of response times. If response time variance is high, for instance, it may be difficult to distinguish between transient performance degradation and a denial of service attack. It is well known that variance reduction techniques can improve the task of anomaly detection [9, 11, 39, 43, 48]. One of the goals of this paper is to assess the impact of the number of servers in a cloud computing environment on the variance of response times and accuracy of anomaly detectors.

We consider a fixed budget to be allocated in cloud premises, aiming at cost-optimal cloud deployments [29, 49]. The budget can be split into several servers or few more powerful servers. We then pose the following questions:

- As the number of servers increases, and the capacity of each of them decreases, how does the mean and the variance of response time vary?
- How does the accuracy of anomaly detection classifiers vary as a function of the number of servers?

While answering the first question, we discovered that both the mean and the variance of the response time first decrease and then increase as the number of servers grows. This, in turn, motivated the second question (Figure 2).

We focus on metrics directly affecting customers without violating the user privacy, e.g., response times. We indicate that those metrics are sensitive to anomalies to the extent that they can then be used to support an anomaly detection framework. As customer-affecting metrics are intrinsically publicly available, sharing them with third-parties responsible for anomaly 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. In addition, anomaly detection approaches based on performance signatures have the potential of detecting zero-day attacks [34], as those approaches are based on detecting performance deviations from regular behavior and do not require detailed knowledge of attack history.

Prior art: Previous work has shown that performance engineering can be used for implementing attack detection methods. For instance, Avritzer et al. [7] propose an approach that leverages performance signatures based on CPU,

---

---
Fig. 2. The considered anomaly detection mechanism aims to detect anomalies solely based on job response times. When is it feasible to identify if the system is under attack solely based on response times? Jobs of types $a$, $b$ and $c$ correspond to short jobs, short jobs impaired by attacks and long attack jobs, respectively. Depending on the system setting, the waiting time of jobs in the waiting queue can be used to distinguish jobs of type $a$ and type $b$. Under which system configurations does the waiting time contain enough information to allow the automatic distinction between jobs of types $a$ and $b$, implying the feasibility of locally detecting anomalies solely based on response times?

Fig. 3. Proposed framework for a performance-based anomaly detection system

I/O, memory and network usage for the detection of security anomalies. The approach has been shown to be very accurate and to outperform, in some cases, anomaly detection systems [34] based on attack history. The architecture of performance and security in cloud computing systems incurs several tradeoffs, as for example, the overhead of implementing security protocols [51], the configuration of database partitioning systems for anomaly detection while optimizing performance [1], and security-related cache invalidation analysis to ensure scalability of data-intensive applications [31]. In this work, we indicate that models can be instrumental to assess the impact of architecture design decisions in the tradeoff landscape. In particular, we evaluate the impact of the number of servers on the mean and variance of response time at the application level, and on the accuracy of anomaly detectors. 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 anomalies and not to the variability in system performance.
Contributions: In summary, our key contributions are twofold:

Analytical model to derive mean and variance of baseline response time: We apply a heavy-tail bimodal modeling approach, where short jobs represent regular transactions and long jobs represent resource consuming transactions, e.g., due to denial of service attacks. In our modeling approach, long jobs are related to anomalies. Using the bi-modal 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). Although the bi-modal model is admittedly simple, it already serves our purposes, namely to illustrate the tradeoffs between performance and predictability (Sections 3-5) and accuracy of anomaly detection (Section 6).

Optimal number of servers: Using the proposed model, we consider the problem of determining the optimal number of servers accounting for scalability and security aspects, e.g., anomaly detection based on average response time.

Figure 3 illustrates the framework using the concepts studied in this paper and is further discussed in the upcoming section.

The outline of the remainder of the paper is as follows. In Section 2 we present a performance-based anomaly detection framework that is composed of (i) analytical 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. Those aspects are treated, in that order, in Sections 3 to 6. Related work and the limitations and broad implications of our results are discussed in Sections 7 and 8. Conclusions and directions for future research appear in Section 9.

2 PROPOSED FRAMEWORK

In this section, we describe the proposed performance-based anomaly detection framework. Figure 3 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 [20] to generate a workload model, e.g., user profile distribution [6]. The obtained workload model is the basis of the proposed analytical model (Section 3) and is used to support the specification of the load levels to be used in the performance testing experiments.

(2) Experiments in a 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 [21] to measure mean and standard deviation of service time, with and without anomalies. The submitted transaction rate is derived from APM tools. Experimental results to calibrate the performance model are presented in Section 4.

(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 4 to characterize a bi-modal distribution. Short jobs correspond to service times of typical users, and long jobs correspond to security attacks. The analytical model results (reported in Section 5) are used to estimate the number of servers that minimizes response time average and variance, as shown in Table 4.

(4) Calibrate the anomaly detection algorithms using the mean and standard deviation of response time for the optimal number of servers. Machine learning algorithms, such as SVM [22], can be used to perform classification between regular and anomalous service times, so as to efficiently detect anomalies at an early stage, soon after
Table 1. Table of notation

| Variable | Description | Comment |
|----------|-------------|---------|
| $\alpha$ | fraction of regular jobs (short jobs) | $\alpha = \lambda_S/\lambda$ |
| $\lambda_S$ | arrival rate of short jobs | $\lambda_S$ |
| $\lambda_L$ | arrival rate of long jobs | $\lambda_L$ |
| $\lambda$ | arrival rate of jobs | $\lambda = \lambda_S + \lambda_L$ |

single server metrics

| Variable | Description | Comment |
|----------|-------------|---------|
| $E(X)$ | service time in single server system | unit: millisecond (ms) |
| $E(X_S)$ | short job service time | $E(X) = aE(X_S) + (1 - a)E(X_L)$ |
| $E(X_L)$ | long job service time | $E(X_L) = 1/E(X)$ |
| $\mu$ | service rate of single server system | $\mu = 1/E(X)$ |
| $\rho$ | utilization | $\rho = \rho_S + \rho_L = \lambda E(X)$ |

multi-server metrics

| Variable | Description | Comment |
|----------|-------------|---------|
| $\mu_K$ | service rate per server in multi-server system | $\mu_K = \mu/K$ |
| $T$ | metric of interest: response time | unit: ms |
| $\sigma^2 = V(T)$ | variance of response time | unit: ms$^2$ |
| $K^*_\mu$ | number of servers that minimizes $E(T)$ | $K^*_\mu = \text{argmin}_K E(T)$ |
| $K^*_\sigma$ | number of servers that minimizes $\sigma$ | $K^*_\sigma = \text{argmin}_K V(T)$ |
| $\mu^*$ | minimum value of $E(T)$ | $\mu^* = \text{min}_K E(T)$ |
| $\sigma^*$ | minimum value of $\sigma$ | $\sigma^* = \text{min}_K V(T)$ |

they occur. Such approach is illustrated in Section 6, indicating how the number of servers $K$ impacts classification performance metrics such as accuracy.

3 ANALYTICAL MODEL

In this section, we describe the proposed analytical model, which extends results by Psounis et al. [37]. 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 regular jobs, and the service time of jobs during an attack period.

3.1 Model description

We consider arrivals that occur according to a Poisson process to a queue served by $K$ servers. When a service completes, the corresponding server starts working on the next job at the head of the line. Under the Kendall notation, such system is referred to as $M/G/K$, where $M$ stands for arrivals that occur according to a memoryless arrival process, and $G$ indicates that they are served by a server with general service time distribution [19, 32]. 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 ([19, Figure 1.7], [17] and [15, Section 7.4]).

Let $\lambda$ be the request arrival rate. Such requests may correspond to typical user requests or attacks.

To characterize the service times of jobs, we begin by considering a single server system. We denote by $E(X)$ the mean service time of requests, assuming a single server system. The mean service times of user requests and attacks are given by $E(X_S)$ and $E(X_L)$, respectively. The subscripts $S$ and $L$ follow from the assumption that user requests are short
Table 2. Table contrasting the performance model from [37] with its security instantiation introduced in this paper

|                         | performance model                  | security model                  |
|-------------------------|------------------------------------|---------------------------------|
| long jobs (highly demanding service) | long tasks                        | attacks/anomalies               |
| utilization, $\rho$     | accounts for regular and long tasks | accounts for regular 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 |

and attacks are long, i.e., $E(X_S) \leq E(X_L)$. Figure 4 summarizes the key elements of the model. Table 1 summarizes the model parameters and notation.

Let $\lambda_S$ and $\lambda_L$ correspond to the arrival rate of regular jobs (short jobs) and attacks (long jobs), respectively. Let $\alpha$ be the fraction of requests corresponding to regular traffic,

$$\lambda = \lambda_S + \lambda_L, \quad \alpha = \lambda_S / \lambda. \quad (1)$$

Then,

$$E(X) = \alpha E(X_S) + (1 - \alpha) E(X_L). \quad (2)$$

We denote by $\rho$ the utilization of the single server system, and by $\rho_S$ and $\rho_L$ the utilization due to short and long jobs. Then,

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

The $M/G/K$ queue is used to characterize response times, i.e., request sojourn times. We denote by $\mu$ the mean service capacity (service rate) of a single server system, measured in jobs served per time unit,

$$\mu = 1/E(X). \quad (4)$$

$$\mu_K = 1/(KE(X))$$

Fig. 4. $M/G/K$ model: the proposed analytical model is used to assess the impact of the number of servers $K$ on standard deviation and mean response times.
We denote by $\mu_K$ the mean service capacity of each server when there are $K$ servers in the system. Note that whereas $\mu$, $E(X_S)$, $E(X_L)$ and $E(X)$ are functionally independent of $K$, $\mu_K$ is a function of $K$.

### 3.1.1 Scaling law
In the remainder of this paper, we consider a fixed budget to be allocated in the deployment of multiple servers. As the number of servers grows, the constant system capacity $\mu$ is subdivided into $K$ servers of capacity $\mu_K$. In particular, we assume that the total system capacity is uniformly distributed among servers,

$$\mu_K = \frac{\mu}{K} = \frac{1}{KE(X)}.$$  \hspace{1cm} (5)

As the number of servers $K$ grows, the mean service time per server linearly increases as $1/\mu_K = KE(X)$.

### 3.1.2 Response time
We denote by $T$ the system response time, our key metric of interest. We leverage results by Psounis et al. [37] 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 time in the waiting queue which is approximated as the waiting time of an $M/G/1$ queue, given by $pE(X_r)/(1 - \rho)$, where $E(X_r)$ is the residual service time. Then,

$$E(T) = KE(X) + p_B(K, \rho_S, \rho_L)E(W|W > 0),$$ \hspace{1cm} (6)

where $E(W)$ is the mean time in the waiting queue, or simply waiting time, and $p_B$ is the blocking probability. To obtain $E(W|W > 0)$, Psounis et al. [37] propose to consider the $M/G/1$ queue as a proxy for the $M/G/K$, indicating the accuracy of the following approximation,

$$E(W|W > 0) \approx \frac{\rho}{1 - \rho} E(X_r) = \frac{\rho}{1 - \rho} \frac{E(X^2)}{2E(X)},$$ \hspace{1cm} (7)

where $E(X_r)$ is the residual life of the service time. We refer the reader to [37] for further details, including the derivation of the blocking probability. Similarly,

$$V(T) = K^2V(X) + V(W|W > 0)p_B(K, \rho_S, \rho_L)$$ \hspace{1cm} (8)

$$\approx K^2E(X^2) + \frac{\rho}{1 - \rho} \frac{E(X^3)}{3E(X)}p_B(K, \rho_S, \rho_L).$$ \hspace{1cm} (9)

$$\sigma(T) = \sqrt{V(T)}$$ \hspace{1cm} (10)

where (9) follows from (8) when the service time of jobs under attack is very long compared against the service time of short jobs. In that case, as argued by Psounis et al. [37, Section 3.4], we have $V(T) \approx E(T^2)$.

Finally, throughout this paper, in our analytical results we also assume that service times of short and long jobs are roughly deterministic,

$$E(X^i) \approx \alpha E(X_S)^i + (1 - \alpha)E(X_L)^i.$$ \hspace{1cm} (11)

The simplifying assumptions above serve to produce plots that are easy to visualize without noise that is inherent to simulations or more refined approximations (contrast Figures 5 and 6 produced with the approximations above against simulation results shown in Figure 7). We conducted a simulation campaign to validate the adopted simplifications, and verified that they serve to extract the essence of our results (see Section 5.4).

### 3.1.3 Model summary
We have introduced the analytical model considered throughout this work: 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 [15], motivating simplifying approximations as suggested by [8, 37]. 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.

The source code to reproduce our results is made available online [42]. It is quite remarkable that a simple and classic model such as the $M/G/K$ still allows us to derive fundamentally novel insights on the tradeoff between predictability and performance (Section 3.3) and that the insights derived from such simple model already serve to guide practitioners (Section 8.1).

### 3.2 Optimal number of servers

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

The tuning of the number of servers $K$ must account for $E(T)$, $V(T)$ and the accuracy of anomaly detectors. Table 2 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 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 (Figure 1).

### 3.3 Key insights

In this section 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 $K$, as far as the gain due to a reduction in blocking probability $p_B$ (second term in (6)) is greater than the corresponding increase in the service time $KE(X)$ (first term in (6)). When the number of servers is large, in contrast, a significant fraction of the servers will be idle, i.e., $p_B \approx 0$, favoring a reduction in the number of servers. In that case, an increase in $K$ will cause an increase in service time $KE(X)$ but produce marginal gains with respect to the blocking probability, as the latter is low at first place (second term in (6)). Together, these two observations imply that there is an optimal number of servers that minimizes mean response time.\footnote{It is well known that, under the considered scaling laws, the mean response time of the $M/M/1$ queueing system is smaller than $M/M/K$, for any $K \geq 2$, irrespectively of whether one considers a single queue or multiple queues [26]. Clearly, when considering more general service times the optimality of the single server system may not hold.}

**Reduction in number of servers favors increased response time predictability**: we found that for all the considered setups there is an optimal number of servers that minimizes the response time variance of a typical job. 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, it may be worth slightly sacrificing mean response time to reduce response time variance.

**Reduction in number of servers favors early detection of attacks based on job response times**: the variability in response 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 response times (see Section 6).
Table 3. Empirical results on response times

|               | Without anomaly | With anomaly |
|---------------|-----------------|--------------|
| Avg. response time | 54.13 ms        | 95.20 ms     |
| Std. deviation of response time | 39.23 ms        | 83.72 ms     |

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 [37, 50]. In this paper, we take a similar approach and leave the formal proof of the conjectures implied by the insights above as a subject for future work.

4 MODEL PARAMETERIZATION: LAB EXPERIMENTS

The parameterization of the proposed model involves the determination of the parameters presented in Table 1. In this paper, we vary $K, \lambda, \alpha$, and $E(X_S)/E(X_L)$ to illustrate their impact on the metrics of interest, which are the mean and the standard deviation of the response times (Section 5) as well as the accuracy of mechanisms of anomaly detection (Section 6).

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 [3, 23]. Different events may cause an increase in response time. In this paper, we focus on security threats, i.e., anomalies due to Mirai botnets. Nonetheless, the framework is general, and can encompass both endogenous and exogenous attack vectors [28]. Future work consists in extending the experiments to account for a wider set of attack vectors.

4.1 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, (ii) a load generator (to generate load to the web-server) [6], 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. Three physical machines were connected to the same Ethernet link: (i) the first machine was set as a web server, working as a virtual store (sockshop), over the Docker platform; (ii) the second machine was set as a baseline load generator, using Apache JMeter with 10 threads and issuing 10 requests per second; (iii) the third machine was set as the attacker, running real malware code from the Mirai botnet (Section 7.3). Its goal is to increase the system load. From the publicly available Mirai source code, we extracted a software module to conduct Mirai attacks to a given target, for a configurable time. Using that module, the machine launched a Mirai DDoS attack, executing 256 threads and using the GET method to overload the web server with small HTTP requests.

https://github.com/microservices-demo/microservices-demo
Fig. 5. Assessing the impact of $\alpha$ on response times, given ratio $E(X_S)/E(X_L) = 0.5686$ as obtained from experiments (Section 4).

4.2 Testbed results

We repeated each of our experiments for five runs. 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 without anomalies, under the setup described in the previous section, we found that the response time mean and the standard deviation equal to $54.13$ ms and $39.23$ ms, respectively. With anomalies, those values increased to $95.20$ ms and $83.72$ ms, respectively.

In our baseline setup the system utilization was low, implying a negligible idle waiting time and blocking probability. Hence, we assume that the response times measured in our experiments without attacks correspond to regular service times (see (6)). 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

$$E(X_S) = 54.13, \quad E(X_L) = 95.20.$$  \hspace{1cm} (12)

All values above are measured in milliseconds (Table 3). 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.

In summary,

1. given $E(X_L)$ and $E(X_S)$ obtained from experiments, with and without Mirai, we parametrize $\rho$ (equation (3));
2. the resulting value of $\rho$ is used to compute $E(T)$ and $V(T)$ as a function of $\alpha$ and $K$ (equations (6) and (8));
3. varying $\alpha$ and the number of servers $K$, we study its impact on the metrics of interest (Figure 5);
4. in addition, we also vary $E(X_L)$ around its experimental estimate to conduct what-if counterfactual model-based analysis accounting for different attack vectors that may have different impact on $E(X_L)$ (Figure 6).

The steps above are illustrated in the following section.
5 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: what are the recommended values of $K$ when the utilization, $\rho$, and the fraction of load that is due to attacks, $1 - \alpha$, are taken into account?

5.1 Numerical results

Figures 5 and 6, together with Tables 4 and 5, respectively, present the numerical results obtained with the analytical model. Solid (resp., dashed) lines in Figures 5 and 6 show the average (resp., standard deviation) of the response time as a function of the number of servers in the system, $K$ (see (6) and (8)).

Figure 5 shows results assuming $E(X_S)/E(X_L) = 0.5686$ as estimated by our experimental results (see Section 4.2). Each subplot, from left to right, corresponds to increasing fraction of jobs due to anomalies, with $1 - \alpha$ varying between 0.01, 0.2 and 0.4. In Figure 6, we maintain $\alpha = 0.99$ and $E(X_S) = 54.13$, and vary $E(X_L)$ so as to reach a target $E(X_S)/E(X_L)$.

5.2 Reducing number of servers favors variance reduction

In most of the scenarios considered, the value of $K$ that minimizes the standard deviation of the response time is smaller than or equal to the value of $K$ that minimizes mean response times (see Figures 5 and 6, and Tables 4 and 5). This suggests that one must trade between performance (response time mean) and predictability (response time variance). A notable exception consists of the setup 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 5.

Smaller values of $K$ make the system more predictable, at the expense of an increase in mean response times. As the fraction of anomalous jobs increases, i.e., as $\alpha$ 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 minimal when $K = 1$ for all the considered scenarios (see Tables 4 and 5).
5.3 Reducing number of servers may be beneficial under attacks

5.3.1 Impact of fraction of anomalous jobs. Next, we consider the impact of the fraction of anomalous jobs, $1 - \alpha$, on response times. Under the model parameterized by our experiments (Figure 5 and Table 4), the fraction of anomalous jobs $1 - \alpha$ did not have a significant impact on the optimal number of servers. In the additional scenarios considered in Table 5, we observe that as $1 - \alpha$ increases the optimal number of servers decreases, irrespectively of whether we account for the mean or the variance as our decision criterion. This is because as $1 - \alpha$ increases, the potential benefits of increasing the number of servers to reduce blocking probability diminish, as the additional servers will more likely be occupied by long jobs.

5.3.2 Impact of utilization. As the utilization increases, the optimal number of servers consistently increases, accounting either for the minimization of the mean or the standard deviation of the response time (see Figures 5 and 6, and Tables 4 and 5). In particular, for low utilization ($\rho = 0.5$), when $\alpha \in \{0.8, 0.6\}$ the optimal number of servers equals 1 or 2 under all the considered scenarios (Figures 5 and 6 and Table 5).

5.4 Model validation through simulations

To check the agreement between the analytical model approximate solution presented in Section 3.1.2 against the $M/G/K$ exact solution, which is not amenable to a simple closed-form expression [17], we conducted a simulation campaign. Our simulations are executed using an efficient simulator which leverages an extension of recursive Lindley equations [25], allowing us to produce numerical results more efficiently than traditional event-driven simulations. The simulations were performed in 5 rounds with 1 million samples per round.

Figure 7 and Table 5 report our simulation results. They show that the proposed model approximations are more accurate for smaller values of $\rho$. When $\rho = 0.80$ and $\rho = 0.50$, model approximations typically have predictive power to determine the optimal values of $K$ that minimize $E(T)$ and $V(T)$ (columns $K^*_\mu$ and $K^*_\sigma$ in Table 5). In addition, for all the considered scenarios, the qualitative behavior of the model and simulations is similar (Figure 7). Note that scenarios wherein $\rho$ is large, e.g., $\rho = 0.95$, are arguably of less relevance, as system administrators will typically provision the system to avoid heavy traffic.

When $K = 1$ the corresponding M/G/1 system admits solutions in closed form (last two columns of Table 5). It is interesting to observe that by deviating from the single server setup both the mean and the variance of the residence time can often be significantly reduced. Nonetheless, the single server system is still the optimal choice in multiple scenarios, e.g., when the utilization is small and attacks are not frequent ($\rho = 0.5$ and $\alpha = 0.6$, as shown in the last line of Table 5).

| Setup | $\rho = 0.5$ | $\rho = 0.8$ | $\rho = 0.95$ |
|-------|-------------|-------------|-------------|
| Fig. 5 | $K^*_\mu$ | $K^*_\mu$ | $K^*_\mu$ |
| $\sigma = 0.99$ | 1 | 1 | 2 |
| $\sigma = 0.80$ | 1 | 1 | 2 |
| $\sigma = 0.60$ | 1 | 1 | 1 |

Table 4. Optimal number of servers minimizing average and standard deviation of response time (setup of Fig. 5, $E(X_S)/E(X_L) = 0.5686$).
We begin by considering an SVM classifier. Each sample to the classifier. Each sample, in turn, comprises a single feature, namely the corresponding job response time.

### 6.1 Accuracy of Anomaly Detection

Next, our goal is to assess the accuracy of machine learning classifiers to detect anomalies. To that aim, we vary the

| $E_S / E_L$ | $\alpha$ | $\rho$ | $K^*_E = \arg \min_{E(T)} E(T)$ | $\mu^* = \min E(T)$ | $\sigma^* = \min \sqrt{V(T)}$ |
|-------------|----------|--------|------------------------|-----------------|-----------------|
| 0.0005      | 0.99     | 0.95   | 11(100)                | 305779.58       | 129216.26       |
| 0.0005      | 0.99     | 0.80   | 30(41)                 | 59155.35        | 6111.62         |
| 0.0005      | 0.99     | 0.50   | 8(9)                   | 13832.47        | 274.68          |
| 0.0005      | 0.80     | 0.95   | 6(2)                   | 181909.29       | 61577.73        |
| 0.0005      | 0.80     | 0.80   | 3(1)                   | 214297.84       | 4113.42         |
| 0.0005      | 0.80     | 0.50   | 1(1)                   | 75717.31        | 505.50          |
| 0.0005      | 0.60     | 0.95   | 4(1)                   | 924227.47       | 16675.28        |
| 0.0005      | 0.60     | 0.80   | 2(1)                   | 259296.47       | 3549.80         |
| 0.0005      | 0.60     | 0.50   | 1(1)                   | 97425.93        | 134.80          |
| 0.0005      | 0.99     | 0.95   | 29(90)                 | 6246.33         | 8566.91         |
| 0.0005      | 0.99     | 0.80   | 12(23)                 | 2850.30         | 120.64          |
| 0.0005      | 0.99     | 0.50   | 5(6)                   | 1132.58         | 29.86           |
| 0.0005      | 0.80     | 0.95   | 7(2)                   | 79372.55        | 3706.02         |
| 0.0005      | 0.80     | 0.80   | 3(1)                   | 21086.50        | 304.30          |
| 0.0005      | 0.80     | 0.50   | 1(1)                   | 7515.90         | 23.16           |
| 0.0005      | 0.60     | 0.95   | 4(1)                   | 91616.80        | 4034.39         |
| 0.0005      | 0.60     | 0.80   | 2(1)                   | 25826.31        | 129.70          |
| 0.0005      | 0.60     | 0.50   | 1(1)                   | 9735.78         | 25.59           |
| 0.0005      | 0.99     | 0.95   | 4(3)                   | 316.11          | 80.60           |
| 0.0005      | 0.99     | 0.80   | 2(3)                   | 221.60          | 7.76            |
| 0.0005      | 0.99     | 0.50   | 2(2)                   | 146.39          | 281.72          |
| 0.0005      | 0.80     | 0.95   | 8(1)                   | 5948.63         | 367.26          |
| 0.0005      | 0.80     | 0.80   | 2(1)                   | 1861.65         | 67.33           |
| 0.0005      | 0.80     | 0.50   | 1(1)                   | 715.42          | 1.66            |
| 0.0005      | 0.60     | 0.95   | 4(1)                   | 8445.50         | 385.45          |
| 0.0005      | 0.60     | 0.80   | 1(1)                   | 2487.21         | 16.69           |
| 0.0005      | 0.60     | 0.50   | 1(1)                   | 970.94          | 3.27            |

Table 5. Optimal number of servers minimizing average and standard deviation of response time ($K^*_E$ and $K^*_V$, resp.). The table also reports the minimum value of the average and standard deviation of response time ($\mu^*$ and $\sigma^*$, resp.), and the metrics for the corresponding M/G/1 system.

### 6 ANOMALY DETECTION

Next, our goal is to assess the accuracy of machine learning classifiers to detect anomalies. To that aim, we vary the number of servers $K$, and report results on the performance of classifiers as a function of $K$. The classifiers, in turn, classify jobs based on whether they had their response times impacted by anomalies as detailed below.

#### 6.1 Accuracy of Anomaly Detection

We begin by considering an SVM classifier.

**Input feature:** A classifier is developed to detect anomalies. Each job that concludes its service produces a new sample to the classifier. Each sample, in turn, comprises a single feature, namely the corresponding job response time.
Response times (i) can be shared among multiple stakeholders, (ii) are readily available, as they can be measured by end users under low overhead, and (iii) have minimal privacy implications. Therefore, we posit that using response time as feature for anomaly detection constitutes one of the simplest privacy-preserving approaches for anomaly detection.

**Target classes:** The classifier classifies each job into one of three classes: (a) short jobs not impaired by anomalies, i.e., short jobs not affected by long jobs, whose sojourn in the system did not overlap with that of a long job; (b) short jobs impaired by anomalies, i.e., short jobs affected by long jobs, whose sojourn in the system overlapped with that of a long job; and (c) long jobs. The three target classes are illustrated in Figure 2.

**Challenges:** Classifying among those three classes solely based on response times is non trivial, as the impairment suffered by short jobs may initially be subtle when anomalies are initiated. As we consider deterministic service times in our simplified model, it is trivial to detect the presence of anomalies after the fact, in retrospect. Nonetheless, it is challenging to detect anomalies in hindsight, after a small set of short jobs have been slightly impaired by the anomaly.

The goal of the classifier is to enable an early detection of anomalies leveraging those subtle but possibly statistically significant response time impairments.

**Training and test sets:** We collected 50,000 samples from our $M/G/K$ simulator to run the experiments. A fraction of 20% of the samples was used for evaluating the SVM classifier (test set) and 80% for training (training set). The whole process was repeated with the number of servers varying from 1 to 100.

**Accuracy:** To assess the classifier performance, we estimate its accuracy under the test set. The accuracy is the fraction of jobs correctly classified in one of the three target classes, divided by the number of jobs classified.
Fig. 8. Classifier’s accuracy as a function of the number of servers, with \( \alpha = 0.99 \).

Fig. 9. Proportion of the number of short jobs that find the system with anomalies, with \( \alpha = 0.99 \).

**Results:** Figure 8 shows how accuracy varies as a function of the number of servers \( K \). We let the server utilization \( \rho \) vary between 0.95, 0.80 and 0.50, and \( \alpha = 0.99 \) (other values of \( \alpha \) were also considered, with similar results).

*The classifier accuracy initially decreases as \( K \) grows:* starting from \( K = 1 \), in most of the considered scenarios both \( E(T) \) and \( V(T) \) decrease as \( K \) grows. Indeed, an increase in \( K \) reduces blocking, which favors better performance (see discussion in Section 3.3). Nonetheless, this increase in performance comes at the cost of a more challenging classification of jobs. When \( K = 1 \), the blocking of shorts jobs is an early signal that the system is facing an anomaly. As \( K \) increases, such blocking decreases, and the early detection of anomalies through the assessment of response times becomes more challenging.

*Further increasing \( K \) favors accuracy, but hurts performance:* as \( K \) is further increased, the impact of the long jobs on the service times of short jobs grows. Then, the accuracy of the classifier increases. Such increase in classification accuracy comes at the cost of performance degradation.

*For large values of \( K \), the classification task is trivial:* when \( K \) is large, most short jobs will be affected by long jobs. Indeed, Figure 9 reports the fraction of short jobs whose sojourn in the system overlapped with at least one anomalous job. As shown in Figure 9, such fraction is close to 1 when \( K \geq 80 \) and \( E(X_S)/E(X_L) \) is either 0.0005 or 0.005. In a system with many small servers, most short jobs will suffer from the impairments of anomalous jobs and the class of short jobs not affected by long jobs vanishes.

**Take away message:** Figures 8 and 9 show that the detection of anomalies is simplified when the number of servers is either small or large. Otherwise, one needs to cope with a tradeoff between response time mean, response time
Fig. 10. Evaluating threshold strategies and feasibility of anomaly detection based on waiting times: (a) threshold strategy; (b) a priori class probabilities and (c) p-values and statistical hypothesis test to verify the feasibility of anomaly detection based on waiting times: as $K$ grows, waiting times decrease and it becomes more challenging to detect anomalies based on those. Recall that response times are waiting times plus service times, and that service times are assumed to be deterministic per class. We let $\rho = 0.5, \alpha = 0.8$ and $E(X_S)/E(X_L) = 0.05$.

variance, and the accuracy of mechanisms for anomaly detection. Those metrics, in turn, are related to fundamental aspects of the system such as its performance, predictability and security.

6.2 Threshold Strategies

Next, we report illustrative results on the anomaly detection thresholds derived from our numerical evaluation. To that aim, we report decision trees (DT) and naive Bayes (NB) classifiers to distinguish between the waiting time of jobs that find the system with anomalies from those that did not find anomalies. Note that whereas in the previous section we considered an SVM classifier, in this section we illustrate our results using DT and NB to indicate that our results are not bound to a specific classifier. The evaluation methodology is the same as the one described in the previous section, except that in this section we consider the whole dataset for training purposes, i.e., for learning the thresholds. In addition, we focus solely on short jobs. If the waiting time of a job is smaller than the learned threshold, the job is classified as being executed in a system without anomalies. Otherwise, the job is tagged as an execution facing an anomaly.

We let $\rho = 0.5, \alpha = 0.8$ and $E(X_S)/E(X_L) = 0.05$, and vary $K$ between 1 and 20. This scenario corresponds to Figures 8(c) and 9(c). The thresholds learned through NB and DT classifiers agreed with each other. As shown in Figure 10(a), under the considered scenarios the anomaly detection thresholds closely followed the mean waiting time of jobs that did not find anomalies in the system.

The dependency of the learned threshold on $K$ is non-trivial, as the threshold is affected by a number of factors, including the fraction of short jobs that find an anomalous job in execution as well as the variance of the waiting times. The former impacts the a priori probability of classifying a job as a short job impaired by an anomaly (Figure 10(b)), and the latter impacts the likelihood of observed waiting times given the target classes. In particular, the imbalance of the number of samples across classes partially explains why the thresholds do not always reside in between the curves corresponding to the average time of short jobs impaired by anomalies and short jobs not impaired by those anomalies. Indeed, the fact that in the considered setup the fraction of short jobs that do not find an anomalous job in execution is below 0.4 (see Figure 10(b)) favors a reduction in the threshold values.
6.3 Feasibility of anomaly detection

Next, we consider the feasibility of detecting anomalies solely based on waiting time or response time measurements, from the standpoint of statistical hypothesis tests. In particular, when the number of servers grows large, the waiting times are negligible, rendering the detection of anomalies based on waiting times unfeasible. Indeed, we conducted statistical hypothesis tests to check if the waiting times of jobs that found an anomaly in the system have the same mean as the waiting times of jobs that did not find anomalies. The null hypothesis $H_0$ corresponds to the two means being equal, and the alternative hypothesis $H_1$ corresponds to the two means being different.

We illustrate our results under the same setup as considered in the previous section. As shown in Figure 10(c), for $K \leq 6$ the p-values obtained from t-tests are negligible (much smaller than 0.05), meaning that the null hypothesis can be rejected and it is feasible to run statistical hypothesis tests to detect anomalies based on waiting times. However, for larger values of $K$ the p-values grow. In the limit when $K = \infty$ we have an $M/G/\infty$, for which the residence times of all jobs are decoupled, i.e., short jobs are not affected by long jobs, waiting times are zero, and it is infeasible to detect the presence of anomalies based on the response time of regular jobs, under the assumptions considered in this work. In those cases, one needs to resort to additional side information for anomaly detection.

7 RELATED WORK

There is a vast literature on analytical and experimental aspects related to security and performance [1, 2, 53]. 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.

7.1 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 [24, 37]. In this paper, we apply the model introduced in [37] to the performance analysis of denial of service attacks in multi-server systems. Security implications of [37] have been briefly pointed out by [40]. In this work, we take a step further, studying how the mean and 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 queueing theory and autonomic computing [45, 46]. The optimal number of servers in $M/G/K$ queues and its variants has also received some attention in the past [38, 44, 50]. 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 evaluating 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, 41]. 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 [19, 32]. 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 analytical model.
7.2 Anomaly detection

In [10] a survey of anomaly detection techniques 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 [18, 27] and [12] 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, 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.

7.3 Mirai

In this paper we launched Mirai DDoS attacks in a controlled lab environment using the Mirai publicly available source code. In [3] the authors describe the Mirai botnet used to create a significant DDoS attack in 2016. This attack harnessed the power of insecure IoT devices. The authors provided an analysis of the Mirai timeline covering seven months, which included up to 600k intrusions. In [23], the authors present a review of Mirai and its mutations and alert to the risks posed by large botnets formed by using compromised IoT devices. The key lesson from Mirai’s attack is that IoT devices are currently a prime target of attacks and can be easily harnessed in large numbers to create a massive bot attack.

7.4 Security-scalability tradeoff

Nylander et al. [36] propose an architecture using a single queue load-balancing front-end combined with admission control 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. In general, the set-up of scalable cloud computing services impacts both performance and system security aspects.

In [30, 31] 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: (i) mandatory encryption of sensitive information, and, (ii) restricting data encryption to encrypt only data that is not scalability affecting. Our work is related to [31], as we also propose 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 [14] 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.

7.5 IDS tools

The literature on IDS tools is vast [13, 16, 34, 35, 52, 52]. A comprehensive survey of IDS tools is presented in [34], 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) [34]. 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. Milenkoski et al. [34] report that one of the most important open research questions in this domain is the development of IDSs for detecting zero-day attacks and advanced persistent threats (APTs) [16]. The anomaly detection approaches based on performance signatures as considered in this paper do not require detailed knowledge of attack history, and for this reason they are suitable to address the challenges associated to zero-day attacks and APTs detections.

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 [4].

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 [5] 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 in this paper can be instrumental to set the baselines of those algorithms.

8 PRACTICAL IMPLICATIONS, ASSUMPTIONS AND LIMITATIONS

8.1 Practical engineering 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.

From the infrastructure setup perspective, we analytically observed that the number of servers that minimizes mean response time is typically larger than or equal to the number of servers that minimizes response time variance. 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.

8.2 Assumptions and limitations

Next, we discuss some of the simplifying assumptions considered in the paper to yield a tractable model, as well as some limitations of the current work.

Model accuracy: the proposed approximations to solve the analytical model allow us to predict the optimal number of servers that minimizes the mean or the variance of response times (Section 5.4). Although there is space for improvement of the approximations, e.g., if the goal is to assess the minimum mean or variance of response times, such refinements are possible at the expense of a more complex solution, which is out of the scope of this work. In this paper, we focus on the simplest closed form expressions (eqs. (6)-(9)) that were able to capture the trends and insights of interest (Section 3.3). For ideas of further refinements of the proposed approximations, we refer the reader to [37].

Optimal number of servers: we have separately considered the problem of minimizing the mean and the variance of response times. To combine the two problems, one may either minimize a weighted sum of the two metrics, or minimize mean response time with constraints on its variance. The target application will determine how to account for the two metrics. Our results indicate that there is a tension between the two competing metrics, and that decreasing variance may come at the cost of an increase in the mean response time.

Practical experiments: the performance of anomaly detection algorithms based on response times relies on reference values for the mean and variance of response times [5]. Section 4 shows how real experiments can be used to parameterize the proposed model, and in Section 6 we report results on how the number of servers impacts anomaly detection algorithms. The model and insights introduced in this paper can be leveraged by anomaly detection algorithms, noting that a further integration of the proposed model and anomaly detection algorithms is left as topic for future research.

9 CONCLUSIONS

Anomaly detection has been widely recognized as an important production tool in several application domains, such as medical systems, banking networks and failure diagnosis in mission-critical domains [10]. In this paper, we have presented an anomaly detection framework that is supported by standard performance engineering processes, 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 initial experiments are encouraging, showing that the customer-affecting metric is sensitive to the security anomalies evaluated.

Our results show 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 response times is a key component that impacts the performance of anomaly detection algorithms, and we believe that this paper is a first step towards bridging the gap between analytical models for performance evaluation and their security implications.

As topics for future research, we envision the application of the proposed framework to large mission-critical systems. Specifically, additional experiments are required to evaluate how to integrate admission control and load balancing [36] with the cloud computing infrastructure configuration approach to ensure mean response time predictability. In addition, we plan to compare the performance of anomaly detection algorithms based on different approaches, such as bucket algorithms [5] which can naturally leverage the statistical characterization of response times proposed in this paper against other machine learning algorithms [22].

REFERENCES

[1] Firas Alomari and Daniel A Menascé. 2012. An autonomic framework for integrating security and quality of service support in databases. In IEEE Sixth Int. Conf. on Software Security and Reliability. IEEE, Gaithersburg, MD, USA, 51–60. doi: 10.1109/SEER.2012.15.

[2] Firas Alomari and Daniel A Menascé. 2014. Efficient response time approximations for multiclass fork and join queues in open and closed queuing networks. IEEE Transactions on Parallel and Distributed Systems 25, 6 (Jun. 2014), 1437–1446. doi: 10.1109/TPDS.2013.70.

[3] Manohar Antonakakis, Tim April, Michael Bailey, Matt Bernhardt, Elie Burstein, Jaime Cochran, Zakir Durumeric, J Alex Halderman, Luca Invernizzi, Michalis Kalitsis, Deepak Kumar, Chat Lever, Zane Ma, Joshua Mason, Damian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas, and Yi Zhou. 2017. Understanding the Mirai botnet. In 26th USENIX Security Symp. USENIX, Vancouver, BC, Canada, 1092–1101.

[4] Alberto Avritzer, Andre Bondi, and Elaine J. Weyuker. 2005. Ensuring Stable Performance for Systems That Degrade. In 5th Int. Workshop on Software and Performance (WOSP). ACM, Palma de Mallorca, Spain, 43–51. doi: 10.1145/1071021.1071026.

[5] Alberto Avritzer, Robert G. Cole, and Elaine J. Weyuker. 2007. Using performance signatures and software rejuvenation for worm mitigation in tactical MANETs. In 6th Int. Workshop on Software and Performance (WOSP). ACM, Buenos Aires, Argentina, 172–180. doi: 10.1145/1216993.1217023.

[6] Alberto Avritzer, Vincenzo Ferme, Andrea Janes, Barbara Russo, Henning Schulz, and André van Hoorn. 2018. A quantitative approach for the assessment of microservice architecture deployment alternatives using automated performance testing. In European Conf. on Software Architecture (ECSA) (LNCS), Vol. 11048. Springer, Madrid, Spain, 159–174. doi: 10.1007/978-3-030-00761-4_11.

[7] Alberto Avritzer, Rajanikanth Tanikella, Kiran James, Robert G Cole, and Elaine J Weyuker. 2010. Monitoring for security intrusion using performance signatures. In Proc. 1st Joint WOSP/SIPEW Int. Conf. on Performance Engineering. ACM, San Jose, CA, USA, 93–104. https://doi.org/10.1145/1712665.1712623.

[8] Shelby L Brumelle. 1971. Some inequalities for parallel-server queues. Operations Research 19, 2 (Apr. 1971), 402–413. doi: 10.1287/opre.19.2.402.

[9] Kevin M Carter and Alfred O Hero III. 2008. Variance reduction with neighborhood smoothing for local intrinsic dimension estimation. In 2008 IEEE Int. Conf. on Acoustics, Speech and Signal Processing. IEEE, Las Vegas, NV, EUA, 3917–3920. doi: 10.1109/ICASSP.2008.4518510.

[10] Varun Chandola, Arindam Banerjee, and Vinip Kumar. 2009. Anomaly detection: A survey. Comput. Surveys 41, 3, Article 15 (Jul. 2009), 58 pages. doi: 10.1145/1541880.1541882.

[11] Jinghua Chen, Satek Sathe, Charu Aggarwal, and Deepak Turaga. 2017. Outlier detection with autoencoder ensembles. In Proc. SIAM Int. Conf. on Data Mining. SIAM, Houston, TX, EUA, 90–98. doi: 10.1137/1.9781611974973.11.

[12] Dorothy E. Denning. 1987. An Intrusion-Detection Model. IEEE Transactions on Software Engineering, SE-13, 2 (Feb 1987), 222–232. doi: 10.1109/TSE.1987.232894.

[13] Muhamad Felemban, Yahya Javed, Jason Kobes, Thamir Qadah, Arif Ghafoor, and Walid Aref. 2018. Design and Evaluation of A Data Partitioning-Based Intrusion Management Architecture for Database Systems. (out 2018). https://arxiv.org/abs/1810.0261v2 .

[14] David Gillman, Yin Lin, Bruce Maggs, and Ramesh K. Sitaraman. 2013. Performance Modeling and Design of Computer Systems. Cambridge University Press, UK.
[20] Christoph Heger, André van Hoorn, Mario Mann, and Dusan Okanovic. 2017. Application Performance Management: State of the Art and Challenges for the Future. In 8th ACM/SPEC on Int. Conf. on Performance Engineering (ICPE). ACM, L'Aquila, Italy, 429–432. doi: 10.1145/3036207.3053674, see also: https://openapm.io/landscape.

[21] Zhen Ming Jiang, Ahmed E Hassan, Gilbert Hamann, and Parminder Flora. 2008. Automatic identification of load testing problems. In Int. Conf. on Software Maintenance (ICSM). IEEE, Beijing, China, 307–316. doi: 10.1109/ICSMA.2008.4658079.

[22] Jaeyeon Jung, Vern Paxson, Arthur W Berger, and Hari Balakrishnan. 2004. Fast portscan detection using sequential hypothesis testing. In IEEE Symp. on Security and Privacy. IEEE, Berkeley, CA, USA, 211–225. doi: 10.1109/SECPRI.2004.1301325.

[23] Georgios Kambourakis, Constantinon Koliais, and Angelos Stavrou. 2017. The Mirai botnet and the IoT Zombie Armies. In IEEE Military Communications Conf. (MILCOM). IEEE, Baltimore, MD, USA, 267–272. doi: 10.1109/MILCOM.2017.8170867.

[24] Matthias Rohr, André van Hoorn, Wilhelm Hasselbring, Marco Lübcke, and Sergej Alekseev. 2010. Workload-intensity-sensitive Timing Behavior Analysis for Fault Localization in Software Systems. In Int. Conf. on Management of Data (SIGMOD). ACM, Chicago, IL, USA, 241–252. doi: 10.1145/1142473.1142501.

[25] Venkat Tadakamalla and Daniel A. Menascé. 2019. Autonomic Elasticity Control for Multi-server Queues under Generic Workload Surges in Cloud Environments. IEEE Transactions on Cloud Computing 14, 8 (Aug. 2019), 1–12. doi: 10.1109/TCC.2020.2992949.

[26] Rufino, Nogueira, Avritzer, Menasché, Russo, Lima, et al.
Improving Predictability of User-Affecting Metrics to Support Anomaly Detection in Cloud Services

[48] Valentina Timčenko and Slavko Gajin. 2017. Ensemble classifiers for supervised anomaly based network intrusion detection. In IEEE Int. Conf. on Intelligent Computer Communication and Processing (ICCP). IEEE, Cluj-Napoca, Romania, 13–19. doi: 10.1109/ICCP.2017.8116977.

[49] Johan Tordsson, Rubén S Montero, Rafael Moreno-Vozmediano, and Ignacio M Llorente. 2012. Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers. Future Generation Computer Systems 28, 2 (2012), 358–367. https://doi.org/10.1016/j.future.2011.07.003 doi: 10.1016/j.future.2011.07.003.

[50] Adam Wierman, Takayuki Osogami, Mor Harchol-Balter, and Alan Scheller-Wolf. 2006. How many servers are best in a dual-priority M/PH/k system? Performance Evaluation 63, 12 (Dec. 2006), 1253–1272. doi: 10.1016/j.peva.2005.12.004.

[51] Katinka Wolter and Philipp Reinecke. 2010. Performance and Security Tradeoff. In 10th Int. School on Formal Methods for the Design of Computer, Communication and Software Systems (LNCS), Vol. 6154. Springer-Verlag, Bertinoro, Italy, 135–167. doi: 10.1007/978-3-642-13678-8_4.

[52] Bruno Bogaz Zarpelão, Rodrigo Sanches Miani, Cláudio Toshio Kawakami, and Sean Carlisto de Alvarenga. 2017. A survey of intrusion detection in Internet of Things. Journal of Network and Computer Applications 84 (Apr. 2017), 25–37. doi: 10.1016/j.jnca.2017.02.009.

[53] Xuancai Zhao, Quizhen Lin, Jianyong Chen, Xiaomin Wang, Jianping Yu, and Zhong Ming. 2016. Optimizing security and quality of service in a Real-time database system using Multi-objective genetic algorithm. Expert Systems with Applications 64 (Dec. 2016), 11–23. doi: 10.1016/j.eswa.2016.07.023.