Security-aware elasticity for NoSQL databases in multi-cloud environments

Athanasios Naskos

Department of Informatics, Aristotle University of Thessaloniki, Greece
E-mail: anaskos@csd.auth.gr

Anastasios Gounaris

Department of Informatics, Aristotle University of Thessaloniki, Greece
E-mail: gounaria@csd.auth.gr

Haralambos Mouratidis

School of Computing, Engineering and Mathematics, University of Brighton, UK
E-mail: H.Mouratidis@brighton.ac.uk

Panagiotis Katsaros

Department of Informatics, Aristotle University of Thessaloniki, Greece
E-mail: katsaros@csd.auth.gr

Abstract: We focus on horizontally scaling NoSQL databases in a cloud environment, in order to meet performance requirements while respecting security constraints. The performance requirements refer to strict latency limits on the query response time. The security requirements are derived from the need to address two specific kinds of threats that exist in cloud databases, namely data leakage, mainly due to malicious activities of actors hosted on the same physical machine, and data loss after one or more node failures. A key feature of our approach is that we account for multiple cloud providers offering resources of different characteristics. We explain that usually there is a trade-off between performance and security requirements and we derive a model checking approach to drive runtime decisions that strike a user-defined balance between them taking into account the infrastructure heterogeneity. Finally, we evaluate our proposal using real traces to prove the effectiveness in configuring the trade-offs.

Keywords: Security-aware elasticity; Horizontal Scaling; Multi-clouds.

Biographical notes: Athanasios Naskos is a Ph.D. candidate in the department of Informatics of the Aristotle University of Thessaloniki. He received his B.Sc (2011) and M.Sc (2013) Diploma in Computer Science from the same institute. His research interests lie in the field of cloud elasticity, distributed data management and model checking.

Anastasios Gounaris is an Assistant Professor at the Dept. of Informatics of the Aristotle University of Thessaloniki, Greece. A. Gounaris received his PhD from the University of Manchester (UK) in 2005. His research interests are in the area of autonomic, adaptive and wide-area data management, massive parallelism, flow and query optimization, data mining and resource scheduling. More details can be found at http://delab.csd.auth.gr/âˆ¼gounaris/

Haralambos (Haris) Mouratidis is Professor of Software Systems Engineering at the School of Computing, Engineering and Mathematics, at the University of Brighton, U.K. He holds a B.Eng. (Hons) from the University of Wales, Swansea (UK), and a M.Sc. and PhD from the University of Sheffield (UK). His research interests lie in the area of secure software systems engineering, requirements engineering, and information systems development. He has published more than 140 papers (h-index 21) and he has secured funding of more than Â£2.4M from national and international funding bodies as well as industrial funding towards his research.

Panagiotis Katsaros is an Assistant Professor at the Dept. of Informatics of the Aristotle University of Thessaloniki, Greece. He has published over 80 research articles in international journals and conference proceedings in the areas of formal analysis, model checking, dependability & security. He has been involved in numerous national and EU projects leading three of them and has open international collaborations with wellknown researchers in the aforementioned areas.

1 Introduction

Cloud computing is an evolving paradigm that has transformed the way organisations and individuals store, share and access their information. It introduces a number of advantages and benefits by supporting a computational infrastructure where availability of resources is dynamic, meaning that hardware and software are provided on demand when users need them at a reasonable monetary cost. On the other hand, the paradigm also creates challenges and introduces concerns related to security. In fact, many organisations and individuals are still avoiding cloud services mostly because they are not sure if the services provided, typically by different providers, are suitable for their security requirements.

Security concerns related to *data leakage* and *data loss* are of particular importance. Simply speaking, data leakage is the unauthorised transfer of data from one user to another. Each user should have access to their own data and not be able to access the data of others unless are authorised to do so. In the cloud, the risk of data leakage is increased due to the storage of data in a multi-tenant environment. A recent study, Grispos et al. (2013) [1], has shown that the risk of data leakage is increased for a company when employees use cloud-based services. On the other hand, data loss refers to a condition where data is destroyed and becomes unavailable. This could be the result of a malicious act (e.g. an attack to an organisation's data), due to human error or due to hardware/software/network failures. In a cloud environment - and in particular in a multi-tenant environment - the risk of data loss can be increased due to the multi-tenancy situation.

We deal with a particular feature of cloud databases, namely elasticity, in light of security concerns. Elasticity allows cloud users to modify the amount of resources used

on-the-fly, so that they can always handle the external request load, even when load changes are unanticipated. It is manifested in three main forms, *horizontal scaling*, where virtual machines (VMs) are added or removed, *vertical scaling*, where the hardware configuration of the existing VMs is modified, and *migration*, where existing VMs are moved between physical hosting machines. More specifically, in this work, we build upon our previous work Naskos et al. (2015) [2] on performance-oriented horizontal scaling so that we can reach elasticity decisions that take into account both performance and security requirements. Performance requirements are expressed as a threshold regarding the maximum allowed response time to user requests, while security requirements are expressed through the probability of data leakage due to multi- tenancy and of data loss through hardware failure and/or due to multi-tenancy. Ideally, one would aim to attain zero violations of the performance threshold, no security incidents, while minimizing the monetary cost associated with the provision of cloud VMs.

1.1 Problem Challenges.

The main challenge in the setting described above stems from the fact that the three requirements, that is bounded response times, minimal monetary cost and protection from failures and data leakage, are essentially intertwined and contradicting to a large extent, as explained below:

- NoSQL databases partition the data across several nodes and can benefit from the inherent feature of cloud infrastructures to dynamically provision resources. The combination of these two characteristics allow cloud databases to horizontally scale when the external load increases, so that more servers become available to respond to user requests. If horizontal scaling is performed carefully, for example, in a load balancing way that avoids over-reacting, the average response time can be maintained to a certain desired level regardless of any changes in the external load. More specifically, more VMs can be added (scale-out) when the load increases, but this comes at an increased monetary cost. Analogously, when the external load decreases, some servers can be released by the user on the grounds that over-provisioned servers incur unnecessary monetary cost. In private clouds, monetary costs are implicit (e.g., through increased energy consumption), whereas, in public clouds, a fee is actually paid. To make matters more complicated, when adding a new server, a transient phase is expected, during which performance does not improve or even may deteriorate, due to data movement to the new server.

- Online services may become unavailable due to failures of both the physical machines and the network, which can lead to data loss. The main mechanism to address this type of threat is through replication (or mirroring) that allows for data to be copied to several servers. The more the copies, the more resistant to failures the system becomes. However, this comes at the expense of higher response times when updating data, since eventually changes need to be propagated to all copies. Moreover, the more VMs are employed, e.g., for performance reasons, the higher the probability a number of VMs equal to or greater than the replication factor to fail thus leading to data loss. This is orthogonal to the fact that the volume of lost data decreases with the number of VMs for the same replication factor.

- Despite any efforts from cloud providers, there is always the danger that malicious cloud users hosted on the same physical machines as the databases get unauthorized

access to data. Intuitively the more physical machines are used to host the database, the higher the danger, whereas, at the same time, public machines are more vulnerable.

To summarize, scaling out a database may improve the performance, but this may incur unnecessary monetary costs due to over-provisioning. Mirroring can be combined with scaling out and may cause performance problems but increases the robustness to failures. Scaling out may also exacerbate the data leakage and data loss threats. As such, keeping latency low through scaling-out is in contrast to monetary cost and avoiding the threat of data leakage and data loss.

1.2 Real-world Motivating Example.

We take motivation for our work from a real-case scenario, the Greek National Gazette Infrastructure, involving the sharing and storage of large number of documents. The Greek National Gazette is responsible for publishing laws and legal decisions on the Government's newspaper in order for these laws and decisions to be active and applicable. Besides legal decisions there are also a number of decision categories originated from the private and public sector that by law must be sent for publications to the Governments's newspaper. In such scenario, the dynamic provision of services with acceptable performance is very important as is the need to make sure that documents are not leaked before the official publication, and they are not lost after they are published. As such, the administrators face the following dilemma: *to temporarily acquire additional and potentially unsafe cloud VMs or to sacrifice performance during peak user request periods?*

1.3 Contributions and Structure.

The contributions of this work are threefold. First, we present a Markov Decision Process (MDP) modelling approach to cloud elasticity in an homogeneous, single cloud provider setting. Our approach is coupled with probabilistic model checking and accompanied by a security threat-aware decision mechanism; to this end, we build upon our performance-oriented proposal in Naskos et al. (2015) [2]. The elasticity decision mechanism can account for user-defined trade-offs between performance and security requirements, while aiming to avoid over- and under-provisioning in any case. Second, we introduce a novel MDP model that accounts for multiple, heterogeneous cloud providers. Third, we present an evaluation that sheds light upon the impact of security requirements on the elasticity behavior. Our results show that our decision making proposal can effectively strike a configurable balance between the conflicting requirements mentioned above. This work is an extended version of the conference paper in Naskos et al. (2015) [3], which focuses only on the single cloud provider setting.

The remainder of this paper is structured as follows. In Sec. 2, we provide the specifications of our models. In Sec. 3, we present the MDP models and the decision mechanisms developed for the single provider setting. In Sec. 4, we introduce the complete approach for multiple and heterogeneous cloud infrastructures. In Sec. 5, we evaluate our proposal for a wide range of security attack and failure probabilities using real cloud database traces. We discuss the related work in Sec. 7 and conclude in Sec. 8. Compared to the conference version in Naskos et al. (2015) [3], the material in Sec. 2 and 4 is new, while the experiments in Sec. 5 are repeated from scratch and Sec. 3 has been revised.

2 Specification of Model Features

Our elasticity policies are based on advanced analysis (i.e., probabilistic model checking) of MPD models. Obviously, the model should be designed in a way that all the essential aspects of elasticity in our problem our captured, so that its analysis leads to good runtime decisions. MDPs are specified by their states, actions, transition probabilities and rewards Puterman (1994) [4]. Below, we provide the list of the main design requirements:

R1: Horizontal Scaling. The model should be capable of capturing the behaviour of the system under different numbers of VMs employed and the transitions between states with different number of VMs.

R2: Performance Uncertainty and Transient Periods. NoSQL systems are particularly complex and it is extremely difficult to derive analytical models that describe their behavior in terms of performance accurately. Moreover, their behaviour is unpredictable and may vary significantly even for the same external conditions. This uncertaintly need to be reflected on the model. Furthermore, during transition from one state to another in terms of the number of VMs employed, the system typically experiences a non-stable transient period, which also needs to be captured by the model.

R3: Security Incidents. Security incidents, along with their probability of occurrence, need to be explicitly covered.

R4: Multi-objective Rewards. Analysis of MDPs heavily relies on the rewards associated with the model entities. To be able to take security-aware elasticity decisions based on such an analysis, the rewards should consider both performance- and security-related incidents (either explicitly or implicitly).

R5: Heterogeneity. The model should differentiate the system's state according to the exact combinations of cloud providers that provide the VMs used.

3 Model-based security-aware elasticity for a single cloud provider

In this section, we first introduce the basic modelling representation at a conceptual level and how it is used to drive performance-oriented elasticity (initially proposed in Naskos et al. (2015) [2]); later in the section, this approach is extended and refined to cover both performance and security issues.

In our initial model, each state corresponds to a different cluster size, where the size equals to the number of active cloud virtual machines (VMs), running a NoSQL database, such as HBase and Cassandra. The NoSQL database is typically both shared and replicated; i.e., its tables are horizontally fragmented and each fragment is allocated to multiple VMs.

Figure 1 introduces a simplified representation of our MDP state space and the enabled actions in each of the shown states. Every state s_i corresponds to the number of VMs that compose the application cluster (e.g. the NoSQL cluster used in Naskos et al. (2015) [2]) with i ($min\ \#VMs \leq i \leq max\ \#VMs$) representing the cluster's size at some time instant.

This illustration of the state space is separated in time sections $(t, t+1, t+2, ...)$ with each one corresponding to a distinct decision step of the cloud provisioning policy. Decision steps are distinct time periods captured in the model; every decision step in the actual

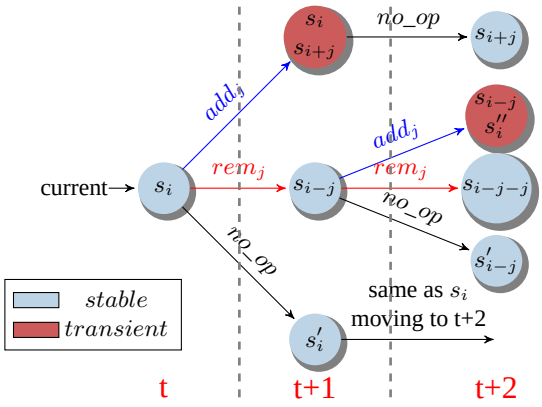


Figure 1 MDP model overview.

deployment, corresponds to pre-specified time periods (e.g. 5 minutes for the present work). We can thus take into account the evolution of the conditions with time, which is particularly important when a decision policy is coupled with external load prediction. As shown in Figure 1, on every decision step the possible elastic actions are *add*, for adding VMs to the cluster, *remove*, for removing VMs and *no_op*, for maintaining the same number of active VMs. The first two actions are also parameterized with the number of VMs added or removed, respectively.

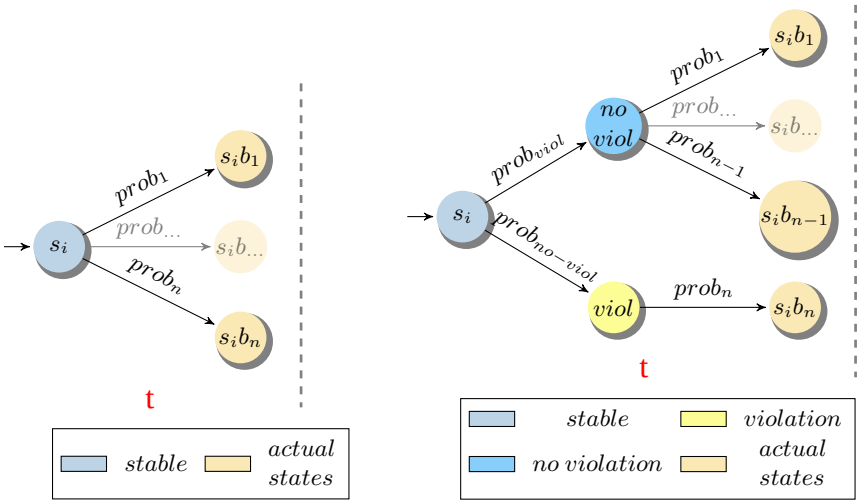
After *add* elastic actions, the decision maker may be idle for a pre-specified time period (e.g. one decision step) to allow the system to stabilize. During this transient period, as the number of active VMs is changed, new VMs need to be (i) created, (ii) booted, (iii) configured, (iv) added to the NoSQL cluster and (v) initialized with data. In Figure 1, the states in the form $\binom{s_i}{s_{i+j}}$ at *t+1* represent *transient* states, i.e. unstable system states due to a recent change in the number of active VMs. Thus, based on the enabled actions at t, we have three states at $t+1$ including two *stable* states s_{i-j} and s'_i, where the number of VMs is not changed, and one transient state. States s_i and s'_i represent a configuration with i VMs, however as the environment evolves, these two states can behave differently to the incoming load (e.g. they may receive different incoming load and may be characterized by different response latency). Also, as we observe, after the s'_i state, the same pattern is repeated with different time sections and state naming conventions, with s'_i now being the current state.

Overall, this model meets the **R1** and the second part (i.e. transient states) of **R2** requirements.

3.1 Model-based elasticity for performance

A common performance requirement is the latency (lat) of processing user requests, i.e. the time elapsed from query submission to answer, not to exceed a certain threshold x, regardless of the number of concurrent users. However, for the same number of VMs and the same amount of incoming load λ, the latency may vary significantly, due to factors that are both external to our model and hard to model; e.g., a time-consuming operating system process is initialized.

To ameliorate this, the probabilistic nature of our model can easily capture the uncertainty of the environment that follows every elasticity decision. The model's



(a) Mapping to actual states (b) The VC modelling approach

Figure 2 Resolution of performance uncertainty

representation in Figure 1 is further elaborated in Figure 2a to account for the possible variability in the application's performance for a given external load and cluster size. More specifically, the conceptual states s_i in Figure 1 correspond to n actual states (shown as s_ib_m in Figure 2a , $1 \leq m \leq n$), to better map the behaviour (b_m) of interest (i.e. performance, security). Each new extended state corresponds to a different expected system behaviour and is derived through clustering the collected and predicted log entries of the past, current and future measurements, for the same external load and cluster size, resulting in deviations from the expected behaviour. The probability of transition to each possible state is proportional to the probability of occurrence of the corresponding state behaviour. With this state transformation our model fully meets the **R2** requirement.

In Naskos et al. (2015) [2], several elasticity policies are examined, and the most effective one was termed as *ADV+VC+PRE*, standing for *advanced+violation-cluster+prediction*. More specifically, the policy is termed as advanced because it computes the *cumulative* reward after a pre-specified number of transitions in the model, called *steps*; this configurable parameter is set to 4 for the current work, based on experimentation with different values. The *VC* label indicates that all the measurements for a given number of VMs and external load that do not meet the latency threshold, are gathered to the same state and all the other measurements, are clustered to more than one states, representing all the viable behaviours of our system. Thus, two new conceptual states are introduced and presented in Figure 2b, the *no violation* state, which is further extended to more than one states and the *violation* state, which is connected with a single state, which depicts the undesirable behaviour of our NoSQL cluster. *PRE* indicates that a prediction mechanism of future incoming load is utilized (i.e. Linear Regression (LR) for the present work). Utilizing the future incoming load prediction, we are able to compute the future latency measurements based on the logged measurements.

Rewards are associated only to model states (action rewards can also be used, however are not considered in the present work) and are derived according to the following utility function:

$$u(vms) = \begin{cases} 0, & \text{if } lat > x \\ 1 + (1/vms), & \text{if } lat \leq x \end{cases} \quad (1)$$

where *vms* is the current number of VMs. As such, this utility function includes a user-specified constraint and manages to take into account both performance issues (through the *lat* threshold) and monetary costs. The latter are implicitly considered by decreasing the utility in a way inversely proportional to the number of machines when there is no performance violation. Overall, this utility function penalizes both under-provisioning and over-provisioning.

Runtime decisions are taken as follows. Every time a decision needs to be taken, the model template described above is dynamically instantiated according to the latest log measurements. Then, a two-phase model verification procedure takes place to decide the optimal path (i.e. finite sequences of states and actions *add/remove/no_op*). To this end, the PRISM tool Kwiatkowska et al. (2009) [5] and its property specification language (PCTL probabilistic temporal logic) are used. In the first phase, we ask for the maximum cumulative reward of the model. I.e., on every transition in the model (i.e., a step in a path), the selected utility function is evaluated and the result is summed to the total reward of the path. In this way, single or multiple optimal paths that lead to the same optimal reward are generated. The PCTL property used to ask for the maximum reward is $Rmax =?\ F[steps = max\_steps]$, where max_steps defines the depth of the verification (i.e., length of the paths) and is set by the user. Secondly, if there are more than one optimal paths, every first action of every optimal path is checked with another PCTL property $Pmax =?\ F[steps = max\_steps\ \&\ violation]$ to define its maximum probability of performance specific Service-Level Agreement (SLA) violation. The first action with the lowest maximum performance violation probability is the one selected by our decision mechanism.

3.2 Model-based elasticity for data leakage

The performance-oriented model aims to avoid performance violations, while avoiding costly over-provisioning. Here, we describe how our model is enhanced with capabilities to capture data leakages and consider them during elasticity decision making thus meeting requirements **R3** and **R4** as well. The modifications refer to both the main model and the decision policy.

More specifically, we further extend the state transformation presented in Figure 2b yielding a hierarchical conceptual model. The new extension is presented in Figure 3. Hence, every s_i state is further connected to one of $safe$ or $not\ safe$ states, where the former stands for no data leakage incident, while the latter denotes the opposite. Hence, the probability of the $s_i b_m$ state is computed through the multiplication of the $prob_m$ probability and the probability of no attack $prob_{i_{np\ attack}}$ or attack $prob_{i_{attack}}$, respectively, since the data leakage attacks and latency violations are considered to be independent events. We consider that there is an explicit mechanism to count and report the number of attacks leading to data leakages in a periodic manner, e.g. Papadimitriou et al. (2011) [6]. The data leakage probability information is used in our models to initialize the transition probabilities to

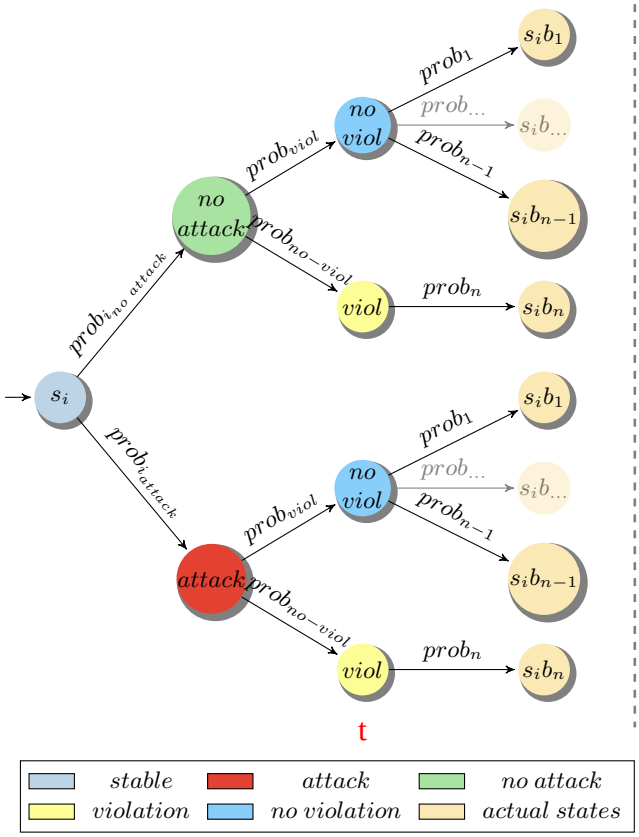


Figure 3 Detailed MDP model stable states.

states that represent *safe* or *not safe* states. A reasonable assumption is that the probability of attacks per VM is the same and equal to $prob_{attack}$, and the attacks on different VMs are statistically independent; in that case, $prob_{i_{attack}}$ becomes equal to $1-(1-prob_{attack})^i$.

In addition, we apply modifications to the performance-oriented model verification procedure and we further employ two additional utility functions for the reward specification. In our first approach, and in order to account for data leakages and performance trade-offs, we propose a 3-parameter function as follows:

$$u(vms) = \begin{cases} 0, & \text{if } attack = true \\ a, & \text{if } lat > x \wedge attack = false. \\ b + (c/vms), & \text{if } lat \leq x \wedge attack = false. \end{cases} \quad (2)$$

where a, b and c are user defined values and $attack$ is a flag that indicates a data leakage. In Sec. 5 we show how setting the 3 parameters, can yield configurable trade-offs between the different objectives, i.e., between (i) security, (ii) performance (in terms of latency violation and under-provisioning), and (iii) monetary cost through avoiding over-provisioning.

The second utility function uses a different weighting scheme between the goals we are trying to achieve and alleviates the need for measurement threshold specification:

$$u(vms) = k \cdot \tilde{p}_{vms_{attack}} + m \cdot \tilde{vms} + n \cdot \tilde{lat} \quad (3)$$

where k, m and n are user defined weights with $k+m+n=1$, $\tilde{p}_{vms_{attack}}$ is the normalized probability of attack for the given number of VMs, $\tilde{vms}$ is the normalized number of VMs and $\tilde{lat}$ is the normalized response latency. Regarding the probability of attack, in our cases, the low and upper bounds are $1-(1-prob_{attack})^{i_{min}}$ and $1-(1-prob_{attack})^{i_{max}}$, respectively; we normalize this interval to $[0,1]$. Similarly, assuming that we know the minimum and maximum number of VMs that can be employed, we normalize the number of VMs to $[0,1]$. For the response latency, where there is no upper bound, we use z-score normalization; then we transform the $[-1,1]$ range into $[0,1]$, while values lower than -1 (resp. greater than 1) are mapped to 0 (resp. 1). Note that this utility function should be minimized rather than maximized.

For both utility functions, the second PCTL property (Sec. 3.1) is transformed to seek the first action with the lowest maximum probability of both performance-specific SLA violation and data leakage in cases of multiple optimal paths.

4 Model-based security-aware elasticity for multiple cloud providers

Our model is further extended to account for multiple cloud providers, hence to meet the **R5** requirement. With this extension, VM instances from different cloud providers with similar performance behaviour and different data leakage probability are supported (note that in general, the performance of similar VM instances provided by different cloud providers can vary Jiang et al. (2009) [7]). Having multiple VM instances with different attack probabilities offered by k cloud providers cp, the attack probability $p_{i_{attack}}$ is computed as

$$p_{i_{attack}} = 1-(1-prob_{attack_{cp_1}})^{\#VMi1_{cp_1}} * \cdots * (1-prob_{attack_{cp_k}})^{\#VMik_{cp_k}}$$

where $prob_{attack_{cp_j}}$ and $\#VMij_{cp_j}$ are the attack probability of a single VM and the number of active VMs, for the cloud provider j (cp_j), respectively; also $i1+i2+\ldots+ik=i$.

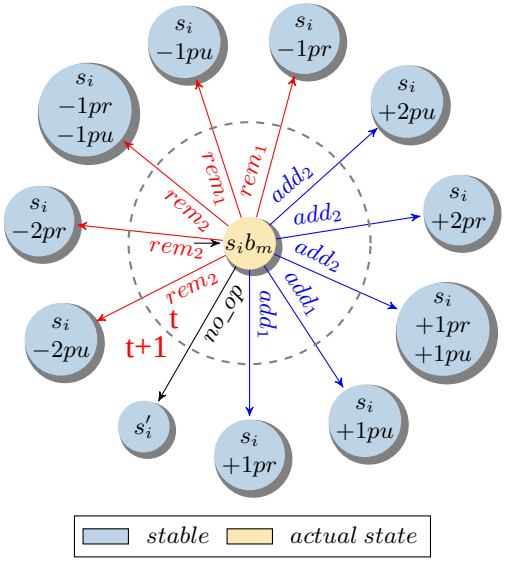


Figure 4 Detailed MDP model states for multi-cloud.

Overall, if there are k providers, adding j VMs does not lead to a single s_{i+j} conceptual state (omitting the transient state for simplicity), but to $\binom{k+j-1}{j}$ possible states. For example, suppose that there are two providers, offering (i) *private VM*, depicted as pr and (ii) *public VM*, depicted as pu in Figure 4. Adding two VMs from state $s_i b_m$ may lead to one of the $\binom{2+2-1}{2} = 3$: (i) $s_i + 1pr + 1pu$, (ii) $s_i + 2pr$, (iii) $s_i + 2pu$ states. Figure 4 shows the extensions to the model for two time sections.

The model solver is responsible to handle the non-determinism and select one of these states. Apparently, this extension further augments the complexity of the verification process. However, the PRISM model checker is able to handle far more complex models; e.g., it verifies models of systems with similar setup to the one presented in Section 5 with up to 10K states approximately, in just a few seconds.

5 Evaluation

5.1 Experimental Setup

We have used logs from a real Cassandra infrastructure to conduct systematic experiments. The collected measurements are used firstly, to populate the initial logs, and secondly, to emulate a real situation. Through emulation, we have managed to fairly test each policy or configuration on an equal basis. The workload consists of asynchronous read requests (req), the volume of which evolves in a sinusoidal manner varying from 4000 to 16000 req/sec coupled with with 2 plateau periods at 13000 req/sec for 1000 time units each. We collected measurements every 30 secs and, in our emulation, a time unit is equal to this measurement collection period. In each sine period, there are 360 measurements. The period of the decision making should be configured according to the volatility of the incoming load of the system and the monitoring frequency. There are cloud providers (e.g. Amazon EC2) that charge extra fees for less than 5 mins monitoring frequency. We allow an elasticity

action to take place every 10 time units, to emulate a system that may modify the VMs every 5 mins (or 10 mins is cases of add action, to allow the system to stabilize). Additions incur a higher transient period due to the higher overhead to create and boot the VM, setup a NoSQL instance and perform data transfer. Scaling-down is simpler as the VMs can be removed immediately (i.e. there is no need for graceful removal of a VM if the replication factor is not affected). As the emulated load is generated based on the logs, which also act as training set, we consider that the system is well trained, and as such, the MDP models are instantiated in an accurate manner. In every up-scale action, up to 3 VMs can be added, while during down-scaling, up to 2 VMs are allowed to be removed in a single step. The cluster sizes varies from 8 up to 18 VMs. Every experiment runs for 3 iterations. Further details are provided in Naskos et al. (2015) [2].

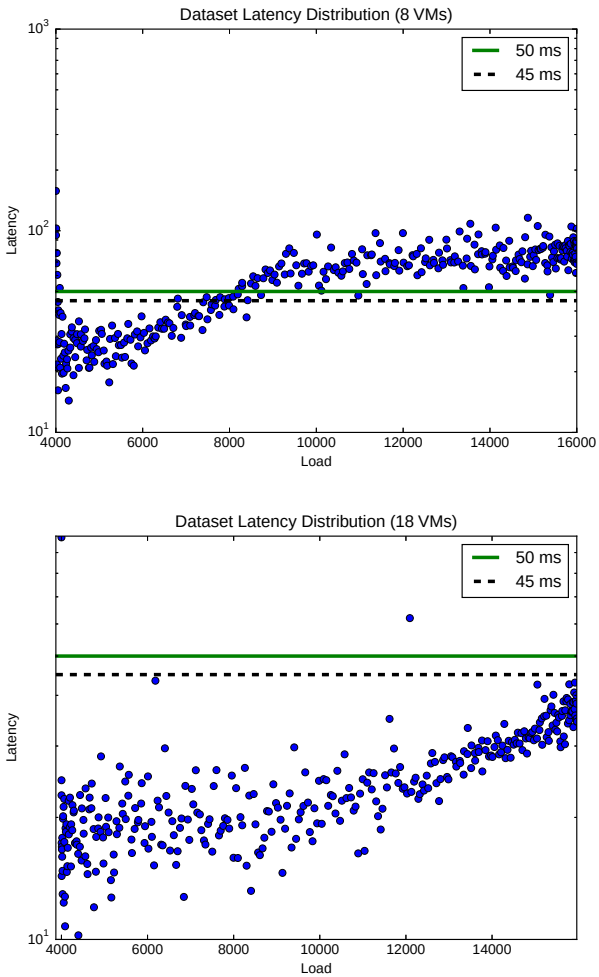


Figure 5 Latencies for 8 (top) and 18 (bottom) VMs

Fig. 5 presents the latency distribution in two characteristic states of the collected dataset, where the dotted line shows the latency thresholds used in the experiments.

5.2 Experimental Results in a Hybrid Cloud Environment

Our experiments show the trade-offs between security attacks and latency violations for a series of utility function configurations and probabilities of attack incidents. We present a setting, where the cloud deployment is hybrid with both private and public VMs. The attack probability for the private VMs is set to 0%, while, for the public one, is 1% for the first set of experiments and 0.1% for the second. Firstly, we present the elasticity behaviour when the extra machines are provided by the public provider exclusively. Later, we present results when the additional VMs can be provided by both parties. For brevity, we present results only for data leakage; the results for data loss are similar, as reported in Naskos et al. (2015) [3].

5.2.1 Data Leakage Attacks - Single Cloud Provider for Extra Machines

In this set of experiments we compare the security-aware model against the baseline model in Section 3.1. For the security-aware model, we employ the utility function in Eq. (2) in five different setups as shown in Table 1. Intuitively, *DLeak-0* tries to avoid attacks at any performance cost. The next three policies, i.e. DLeak-[1-3], place more importance on latency violations than *DLeak-0*. DLeak-4 tries to balance performance and security, emphasizing on attack avoidance slightly less than the DLeak-0 policy. The latency threshold is set to either 45 or 50 msecs.

	DLeak-0	*DLeak-1*	*DLeak-2*	*DLeak-3*	*DLeak-4*
a	100	0.5	100	100	100
b	100	1	100	1000	100
c	1	1	160	1600	16

Table 1 Parameter setup for the utility function in Eq. (2))

In Figure 6, we present the adaptation of the number of VMs to the incoming load for each policy. The red dotted line represents the incoming load while the solid blue line represents the number of active VMs. Except few instabilities, due to the inherent environment uncertainty infused in our emulations, the *ADV+VC+PRE* and DLeak-[1-3] policies can broadly follow the load variation. The DLeak-0 and DLeak-4 policies keep the number of active VM to the most safe state, which is 8 VMs.

	ADV+VC+PRE	*DLeak-0*	*DLeak-1*	*DLeak-2*	*DLeak-3*	*DLeak-4*
45 msecs	12.4	8	12.1	11.3	12.2	8
50 msecs	12.1	8	11.6	10.9	11.7	8

Table 2 Average number of active VMs (1% attack probability)

Initially, we set the probability of data leakage attack per VM per step to 1%; later, we examine data leakage probability of 0.1% that differs by an order of magnitude. The

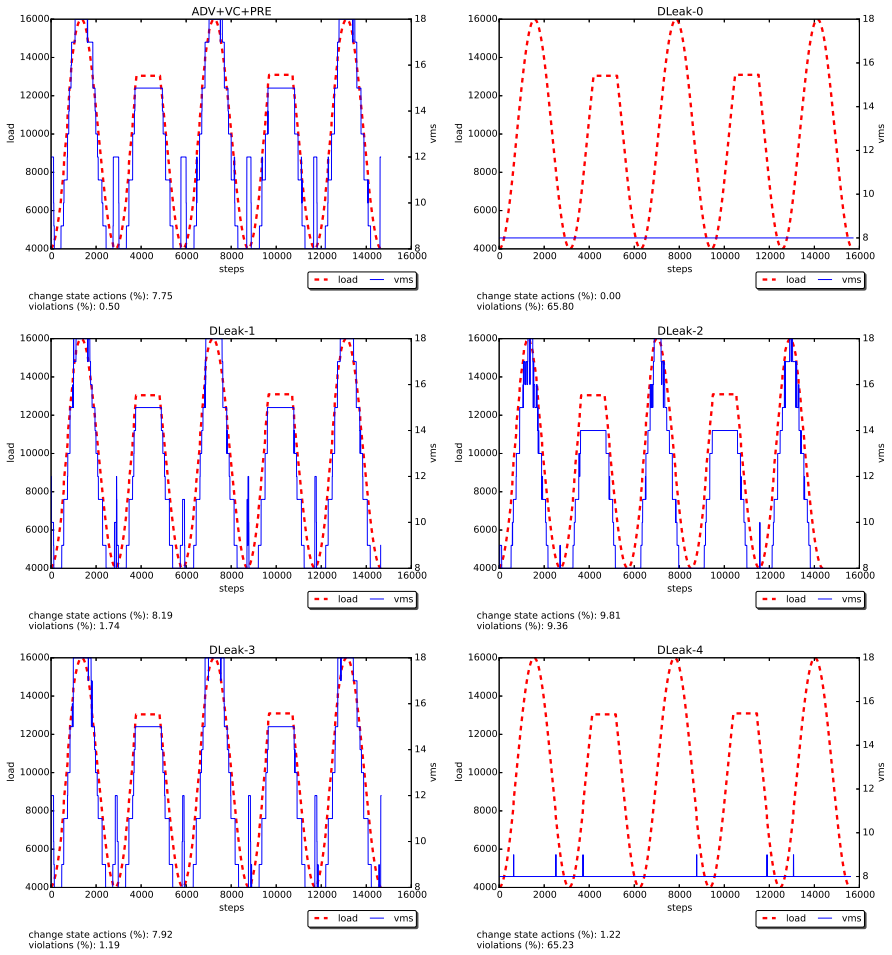


Figure 6 Variation of the external load and the number of active VMs

leftmost pair of columns in Figure 7 (top) presents the percentage of time steps where latency violations (left blue bar) and data leakages (right green bar) occur for the *ADV+VC+PRE* policy. In this experiment, the latency threshold is 45 msecs, and, for cluster size from 8 to 18 VMs, the attack probability ranges from 0% (lower bound) to 9.5% (upper bound). *ADV+VC+PRE* manages to yield a very low number of performance violations at the expense of non-negligible security attacks. The second and sixth pair of columns in the same figure present the results for *DLeak-0* and *DLeak-4*, respectively, where the system is essentially penalized only for the attack situations, as the latency violation reward is very close to the no-attack no-violation case. As expected, the number of VMs is kept low (see Table 2). Overall, the attacks are reduced to their minimum, however the latency violations are reaching their highest percentage (65.8% and 65.23% respectively).

Table 2 needs to be examined in parallel with Figure 7. As we observe in this figure, the *DLeak-2* parameterisation achieves a reduction in the deviation from the lower bound of probability attacks of 30% (from 4.8% to 3.36%) compared to the *ADV+VC+PRE* policy, at the expense of an increase in the latency violations, since the system is prohibited to

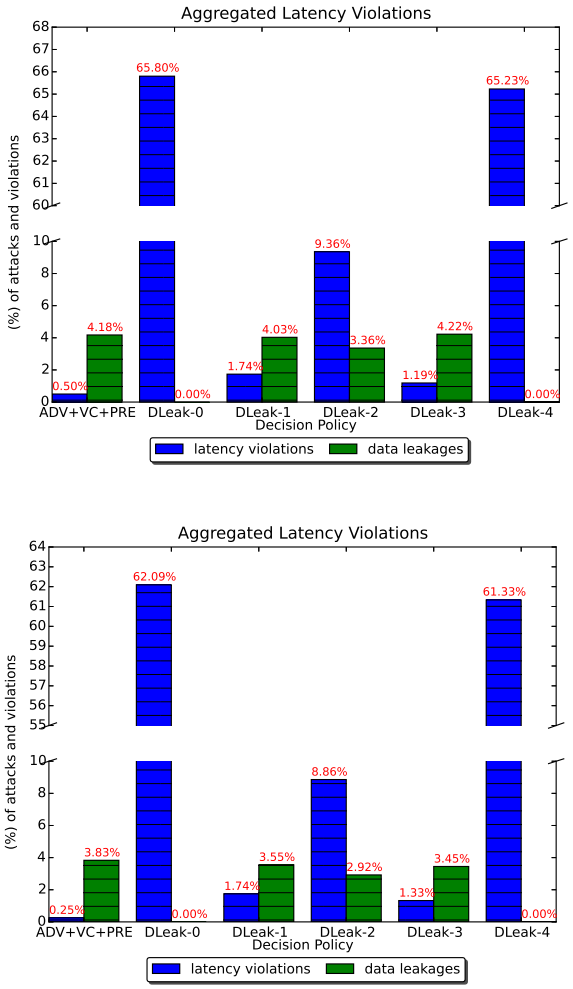


Figure 7 Aggregated Latency Violations and Data Leakage Percentage for 45 msecs (top) and 50 msecs (bottom) latency thresholds and 1% data leakage probability per VM.

scale in several cases to avoid data leakage attacks. *DLeak-1* slightly increases the number of violations (from 0.5% to 1.74%) with a negligible decrease in the data leakage attacks (3.5%). *DLeak-3* parameter setup increases the number of violations without being able to decrease the number of data leakages. As we observe in Table 2, *DLeak-2* keeps the number of active VMs lower than the *DLeak-1* and *DLeak-3*, which explains the decrease in the number of data leakages. This also is an indication that different parameter configurations can achieve different trade-offs.

Fig. 7 (bottom) repeats the same experiment, but with the latency violation threshold set to 50 msecs. The data leakages percentage is decreased in all the security enhanced policies with *DLeak-2* achieving the best tradeoff this time as well.

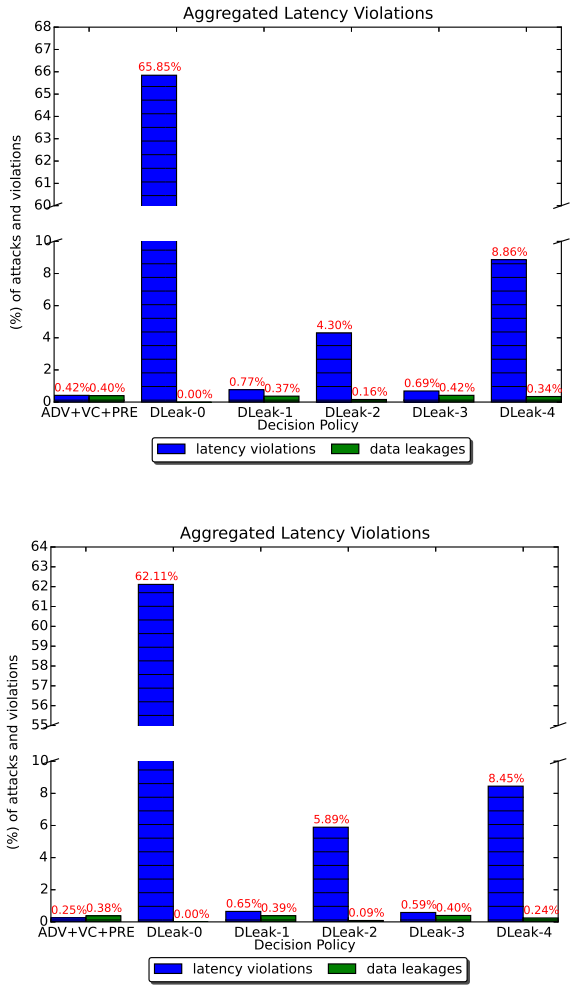


Figure 8 Aggregated Latency Violations and Data Leakage Percentage for 45 msecs (top) and 50 msecs (bottom) latency thresholds and 0.1% data leakage probability per VM.

	ADV+VC+PRE	*DLeak-0*	*DLeak-1*	*DLeak-2*	*DLeak-3*	*DLeak-4*
45 msecs	12.4	8	12.3	11.8	12.3	11.4
50 msecs	12	8	11.8	11	11.9	10.9

Table 3 Average number of active VMs (0.1% attack probability)

In Fig. 8, the data leakage probability because of multi-tenancy is changed to 0.1% per VM per time unit, hence the percentage of data leakage throughout the cluster ranges from 0% to 0.99%. As we observe, the data leakage percentage is reduced by 60% (from 0.4% to 0.16%) for the *DLeak-2* with an increase in the latency violations (i.e. 4.3% from 0.42% achieved by *ADV+VC+PRE* policy), reaching a significantly better trade-off than the other

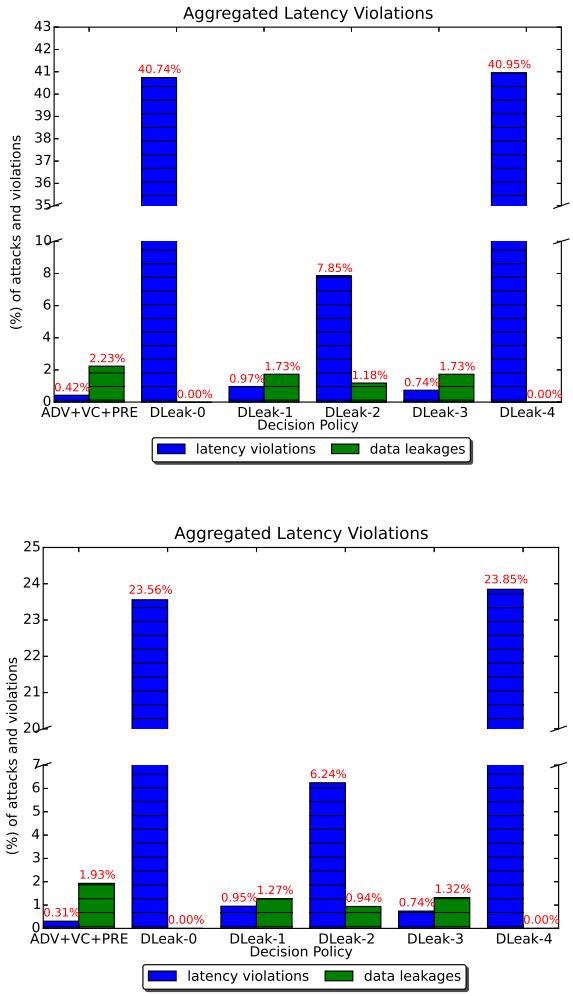


Figure 9 Aggregated Latency Violations and Data Leakage Percentage for 45 msecs (top) and 50 msecs (bottom) latency thresholds and 1% data leakage probability per VM using multiple cloud providers.

setups. The mean number of the active VMs in *DLeak-2* is reduced from 12.4 to 11.8, as presented in Table 3. Interestingly, other configurations, such as *DLeak-4*, fail to reach a beneficial trade-off. When the latency violation threshold is changed to 50 msecs (see Figure 8 (lower)) the same trend applies, with a further reduction of data leakage attacks for the *DLeak-2*, reaching 76% less compared to the *ADV+VC+PRE*.

5.2.2 Data Leakage Attacks - Multiple Cloud Providers for Extra Machines

In this setting, the number of VMs in the private infrastructure ranges from 8 VMs up to 12 VMs and in the public infrastructure from no VMs up to 10 VMs. The total range of VMs is maintained the same with the previous experiments i.e. 8 to 18 VMs, hence the model

solver should choose between a mixture of private and public VMs. The data leakage attack probability per VM per time unit is 1%.

In Figure 9(top), we present experiments with the 45msecs latency threshold. The *ADV+VC+PRE* policy achieved 0.42% of performance violation and 2.23% of data leakage attacks. As it is expected, the data leakage attacks are reduced, compared to the previous experiment of the single cloud provider given that more private VMs can be used. *DLeak-2* managed to drop the data leakage incidents to the half approximately, but with 7.85% latency violations. However, the main difference in the behaviour of the different configurations is that *DLeak-3* achieves an interesting trade-off as well: it reduces security incidents less than *DLeak-2*, but with much fewer performance violations. Also, *DLeak-3* dominates *DLeak-1*. The same trend applies also when the latency threshold becomes 50msecs (Figure 9(bottom)).

5.2.3 Lessons Learned

The main lesson learnt from the above experiments is that the elasticity decision making approach along with the 3-parameter utility function in Sec. 3.2 provides a powerful tool for striking a balance between security and performance requirements. As a rule of thumb to be used by system administrators, we advocate setting the parameters a and b at the order of hundreds (2 orders of magnitude higher than the reward for the security incident) and the parameter c an order of magnitude higher than the maximum cluster size, in order to yield an effective approach in reaching a mid-way balance. Then, if the ratio of data leakage incidents compared to performance ones is considered high, further increasing b and c can be investigated.

5.2.4 Weight-based Utility

	DLeak'-0	*DLeak'-1*	*DLeak'-2*	*DLeak'-3*	*DLeak'-4*	*DLeak'-5*	*DLeak'-6*
k	0	0	0.5	1/3	0.25	0.1	0.3
m	0	0.5	0	1/3	.25	0.25	0
n	1	0.5	0.5	1/3	0.5	0.65	0.7

Table 4 Parameter setup for the utility function in Eq. (3))

We also experimented with the utility function in Eq. 3 for a range of different settings as shown in Table 4. Indicative results are shown in Figure 10. The main observations are that (i) a range of different trade-offs can be achieved through setting the weights accordingly; (ii) these trade-offs are in general inferior to the ones for the previous utility function in terms of ratio of performance and security incidents. The latter is attributed to the fact that the latency threshold is not explicitly taken into account; (iii) given that the goals of avoiding attacks and over-provisioning contradict to the goal of achieving low latency in Eq. 3, setting all the weights to be equal (e.g., as in *DLeak'-3* in Table 4) leads to system under-provisioning, i.e., attacks are avoided at the expense of much severely degraded performance; a more balanced trade-off is accomplished when the *n* weight is set to values > 0.6.

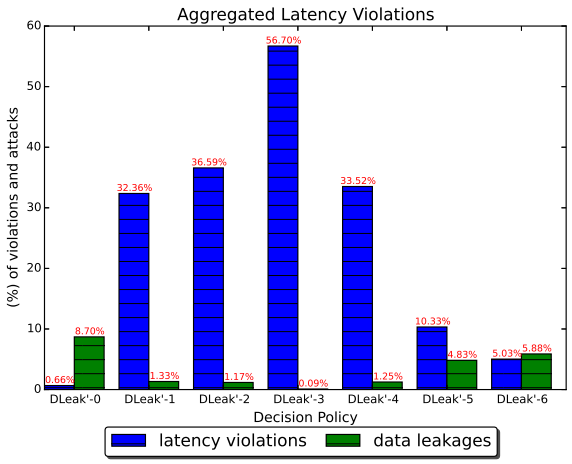


Figure 10 Aggregated Latency Violations and Data Leakage Percentage for 50 msecs latency thresholds and 1% data leakage probability per VM with weighted utility.

6 Real-world Scenario Adaptation

Based on the lessons presented in the Section 5, through our proposal, the Greek National Gazette can become capable of securely utilizing public resources during pick periods, providing performance guarantees. More specifically, the suggestion is that the existing infrastructure is adapted to work in parallel with a public cloud infrastructure (like the Amazon's EC2), so that public VMs can be deployed on the fly when needed. The initial values of maximum number of public VMs, the bounds of elasticity and the latency threshold can be defined by an administrator and adapted at runtime. In addition, the attack probabilities need to be monitored and possibly adapted at runtime (e.g., through online log analysis). Based on our experiments, a good starting point regarding the utility function is to employ Eq. 3 with a setup similar to *DLeak-3*, which is shown to achieve a good balance between security and performance.

7 Related Work

The literature is rich with research efforts that consider security issues within the context of cloud computing. Recent initiatives mainly from the industry and government organisations such as ENISA and Cloud Security Alliance, have sought to produce a number of guidelines and methods to help in the selection of cloud providers as well as addressing some specific security concerns of the cloud. Yet such guidelines appear often too cumbersome with no clear indications as to when a Cloud Service Provider may be considered as not being trustworthy. This makes the valuable information detailed within these documents hard to exploit.

Gong et al. (2010) [8] showed that using a side-channel attack, an attacker can instantiate new VMs of a target virtual machine so that the new VM can potentially monitor the cache hosted on the same physical machine. Mulazzani et al. (2011) [9] showed that attackers can exploit data duplication techniques to access customer data by obtaining hash code

of the stored file. Wenzel et al. (2012) [10] consider security and compliance analysis of outsourcing services in the cloud computing context.

There are also works that focus on the development of model-based approaches to security analysis in cloud environments. A goal-driven approach is introduced to analyse security risks of cloud based system Islam et al. (2012) [11]. Goals, threats and risks are consider from three main components: data, service/application, and technical and organisational measure. We have also contributed to this line of research with the development of a model-based framework that enables elicitation, analysis of security and privacy requirements and selection of deployment models Kalloniatis et al. (2013) [12] and service providers Mouratidis et al. (2013) [13] based on such requirements. These works provide important developments in analysing and modelling security in cloud computing but they do not take into account performance issues.

Our work is also related to proposals that deal with cloud elasticity to maintain specific performance characteristics. Tan et al. (2012) [14] combine cloud elasticity with anomaly prevention, which refers to the resource contention, software bugs or hardware failures. This proposal utilizes a prediction technique based on system metrics to vertically scale the resources of the VMs or to decide for VM migration, i.e. they consider different forms of elasticity, as is also the case in Gong et al. (2010) [15] and Shen et al. (2011) [16]. A work that indirectly solves MDP models utilizing reinforcement learning-based policies to guide elasticity appears in Tsoumakos et al. (2013) [17], which is extended in our previous performance-oriented work in Naskos et al. (2015) [2].

A significant number of proposals use rule-based techniques to guide the elasticity, e.g., Moore et al. (2013) [18] and Copil et al. (2013) [19]. In Copil et al. (2013) [19], a technique is proposed that addresses the implications of an elastic action across multiple dimensions, providing for example the cost implication of a horizontal scaling action. None of those techniques is accompanied by online probabilistic verification of elasticity properties. Finally, model checking and runtime quantitative verification for cloud solutions other than horizontal scaling has been proposed in Calinescu et al. (2011) [20] and Perez et al. (2013) [21]. The former, utilizes PRISM to guide service adaptation, while the latter presents a technique to predict the minimum cost of cloud deployments using PCTL over MDP models. In summary, to the best of our knowledge, our proposal is the first one that addresses the elasticity problem taking into account both performance and security issues.

Finally, several works handle the elasticity between heterogeneous cloud infrastructures, like Copil et al. (2014) [22], Hector et al. (2014) [23] and Qi et al. (2013) [24], or between heterogeneous VM instances of the same cloud infrastructure, like Gupta et al. (2015) [25]. These proposals consider the performance heterogeneity between the different utilized VM instance types. In our proposal, we consider the same performance footprint between the used VM instances and the heterogeneity concerns the different security levels offered by multiple cloud providers. Our current modelling approach is also capable of capturing multiple VM instance types with heterogeneous performance, however this is out of the scope of this paper. None of the aforementioned proposals considers the security heterogeneity between multiple cloud providers, and none of them handles the elasticity using a formalized, dependable approach like the one proposed in this work.

8 Conclusions

This work presents a novel approach to support elasticity decisions for cloud databases, which considers both performance and security requirements. Since, these requirements are contradicting, we have developed a probabilistic model checking solution that accounts for user-defined trade-offs between them and is applicable to multi-cloud environments. As demonstrated by the experiments, our proposal is capable of striking a configurable balance between security-related incidents and performance degradation. Our mechanism can be applied to NoSQL clusters of any size as its scalability is affected only by the scaling size (i.e., the maximum number of VMs that are allowed to add or remove concurrently), which usually does not exceeds a few tens of VMs. Finally, in this work, we have assumed that the attack probabilities are independent. However, if security issues arise due to data transmission between different providers, then the attack probabilities need to become statistically correlated. Our models can be easily support this scenario, since they are orthogonal to the way in which attack probabilities are estimated.

As a future research, we plan to adapt our approach in order to support vertical elasticity. To this end, the envisaged models need to be more fine-grained, considering each VM individually in an extreme case. In general, each different configuration that can be achieved through vertical elasticity need to be treated in a way similar to different cloud providers in this work. This will result in models with much more states and thus is expected to give rise to severe complexity issues that need to be effectively addressed.

References

[1] George Grispos, William Bradley Glisson, and Tim Storer. Using smartphones as a proxy for forensic evidence contained in cloud storage services. *CoRR*, abs/1303.4078, 2013.

[2] Athanasios Naskos, Emmanouela Stachtiari, Anastasios Gounaris, Panagiotis Katsaros, Dimitrios Tsoumakos, Ioannis Konstantinou, and Spyros Sioutas. Dependable horizontal scaling based on probabilistic model checking. In *CCGrid*. IEEE, 2015.

[3] Athanasios Naskos, Anastasios Gounaris, Haralambos Mouratidis, and Panagiotis Katsaros. Security-aware elasticity for nosql databases. In *Model and Data Engineering - 5th International Conference, MEDI 2015, Rhodes, Greece, September 26-28, 2015, Proceedings*, pages 181–197, 2015.

[4] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

[5] Marta Kwiatkowska, Gethin Norman, and David Parker. Prism: probabilistic model checking for performance and reliability analysis. *SIGMETRICS*, 36(4):40–45, 2009.

[6] Panagiotis Papadimitriou and Hector Garcia-Molina. Data leakage detection. *Knowledge and Data Engineering, IEEE Transactions on*, 23(1):51–63, 2011.

[7] Dejun Jiang, Guillaume Pierre, and Chi-Hung Chi. Ec2 performance analysis for resource provisioning of service-oriented applications. In *ICSOC/ServiceWave Workshops*, pages 197–207, 2009.

[8] Chunye Gong, Jie Liu, Qiang Zhang, Haitao Chen, and Zhenghu Gong. The characteristics of cloud computing. In *Proceedings of the 2010 39th International Conference on Parallel Processing Workshops*, ICPPW, pages 275–279, 2010.

[9] Martin Mulazzani, Sebastian Schrittwieser, Manuel Leithner, Markus Huber, and Edgar Weippl. Dark clouds on the horizon: Using cloud storage as attack vector and online slack space. In *USENIX Security Symposium*, 2011.

[10] Sven Wenzel, Christian Wessel, Thorsten Humberg, and Jan Jürjens. Securing processes for outsourcing into the cloud. In *2nd International Conference on Cloud Computing and Services Science*, April 2012.

[11] Shareeful Islam, Haralambos Mouratidis, Christos Kalloniatis, Aleksandar Hudic, and Lorenz Zechner. Model based process to support security and privacy requirements engineering. *IJSSE*, 3(3):1–22, 2012.

[12] Christos Kalloniatis, Haralambos Mouratidis, and Shareeful Islam. Evaluating cloud deployment scenarios based on security and privacy requirements. *Requir. Eng.*, 18(4):299–319, 2013.

[13] Haralambos Mouratidis, Shareeful Islam, Christos Kalloniatis, and Stefanos Gritzalis. A framework to support selection of cloud providers based on security and privacy requirements. *Journal of Systems and Software*, 86(9):2276–2293, 2013.

[14] Yongmin Tan, Hiep Nguyen, Zhiming Shen, Xiaohui Gu, Chitra Venkatramani, and Deepak Rajan. Prepare: Predictive performance anomaly prevention for virtualized cloud systems. In *ICDCS*, pages 285–294, 2012.

[15] Zhenhuan Gong, Xiaohui Gu, and John Wilkes. Press: Predictive elastic resource scaling for cloud systems. In *CNSM*, pages 9–16, 2010.

[16] Zhiming Shen, Sethuraman Subbiah, Xiaohui Gu, and John Wilkes. Cloudscale: Elastic resource scaling for multi-tenant cloud systems. In *SOCC*, pages 5:1–5:14, 2011.

[17] Dimitrios Tsoumakos, Ioannis Konstantinou, Christina Boumpouka, Spyros Sioutas, and Nectarios Koziris. Automated, elastic resource provisioning for nosql clusters using tiramola. In *CCGrid*, pages 34–41, 2013.

[18] Laura Moore, Kathryn Bean, and Tariq Ellahi. A coordinated reactive and predictive approach to cloud elasticity. In *CLOUD COMPUTING*, pages 87–92, 2013.

[19] Georgiana Copil, Daniel Moldovan, Hong Linh Truong, and Schahram Dustdar. Multi-level elasticity control of cloud services. In *ICSOC*, pages 429–436, 2013.

[20] Radu Calinescu, Lars Grunske, Marta Kwiatkowska, Raffaela Mirandola, and Giordano Tamburrelli. Dynamic qos management and optimization in service-based systems. *IEEE Trans. Software Eng.*, 37(3):387–409, 2011.

[21] Diego Perez-Palacin, Radu Calinescu, and José Merseguer. Log2cloud: Log-based prediction of cost-performance trade-offs for cloud deployments. In *ACM SAC*, pages 397–404, 2013.

[22] G. Copil, D. Moldovan, Hong-Linh Truong, and S. Dustdar. On controlling cloud services elasticity in heterogeneous clouds. In *Utility and Cloud Computing (UCC), 2014 IEEE/ACM 7th International Conference on*, pages 573–578, 2014.

[23] Hector Fernandez, Guillaume Pierre, and Thilo Kielmann. Autoscaling web applications in heterogeneous cloud infrastructures. In *IC2E*, 2014.

[24] Qi Zhang, Mohamed Faten Zhani, Raouf Boutaba, and Joseph L. Hellerstein. Harmony: Dynamic heterogeneity-aware resource provisioning in the cloud. In *ICDCS*, pages 510–519, 2013.

[25] Vishal Gupta, Min Lee, and Karsten Schwan. Heterovisor: Exploiting resource heterogeneity to enhance the elasticity of cloud platforms. In *Proceedings of the 11th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, pages 79–92. ACM, 2015.