Anomaly Detection and Root Cause Localization in Virtual Network Functions

Carla Sauvanaud, Kahina Lazri, Mohamed Kaâniche, Karama Kanoun

Anomaly Detection and Root Cause Localization in Virtual Network Functions

Carla Sauvanaud*, Kahina Lazri†, Mohamed Kaâniche*, Karama Kanoun*
*LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France
†Orange Labs, 38 rue du General Leclerc, 92130 Issy-Les-Moulineaux, France
Emails: firstname.name@laas.fr, firstname.name@orange.com

***Abstract*—The maturity of hardware virtualization has motivated Communication Service Providers (CSPs) to apply this paradigm to network services. Virtual Network Functions (VNFs) result from this trend and raise new dependability challenges related to network softwarisation that are still not thoroughly explored. This paper describes a new approach to detect Service Level Agreements (SLAs) violations and preliminary symptoms of SLAs violations. In particular one other major objective of our approach is to help CSP administrators to identify the anomalous VM at the origin of the detected SLA violation, which should enable them to proactively plan for appropriate recovery strategies. To this end, we make use of virtual machine (VM) monitoring data and perform both a per-VM and an ensemble analysis. Our approach includes a supervised machine learning algorithm as well as fault injection tools. The experimental testbed consists of a virtual IP Multimedia Subsystem developed by the Clearwater project. Experimental results show that our approach can achieve high precision and recall, and low false alarm rate and can pinpoint the root anomalous VNF VM causing SLA violations. It can also detect preliminary symptoms of high workloads triggering SLA violations.**

***Keywords*—VNF, monitoring data, machine learning, fault injection, SLA, root cause analysis.**

I. Introduction

Network functions (NFs) were for a long time inseparable of specialized proprietary hardware and stringent topology requirements. With the maturity of virtualization technologies, Communication Service Providers (CSPs) led multiple initiatives to extend hardware virtualization to network services. Moreover, as cloud computing emerged, it became attractive to host NFs as virtual machines (VMs) running on virtualized and well orchestrated commodity servers. These initiatives gave rise to the Network Function Virtualization paradigm (NFV). NFV promises operating expenditure savings, quick deployments, scalability and high flexibility in the management of virtualized NFs. In this context, Virtual Network Functions (VNFs) represent any virtual execution environment configured to provide a given network service. VNFs are often divided into several components each one hosted on single VMs.

As a result of their softwarization, NFs are exposed to anomalies present in IT applications in addition to other categories of anomalies specific to virtualized infrastructures. Unfortunately, NFs are more performance stringent than common IT applications, and anomalies may quickly lead to customers' quality of service degradation. The need of anomaly detection techniques for VNF is consequently preeminent.

Anomaly detection is an important problem that has been widely studied in several domains and often tackled with machine learning approaches [1] [2] [3] [4] [5] [6]. Our goal is to explore the ability of such methods to accurately detect anomalies in VNFs that could lead to performance degradation impacting the end user experience. In more details, this work particularly studies the agreements on the VNF service level contracted between end users and CSP. It aims at detecting anomalies leading the VNF not to achieve these service level agreements (SLAs). As an agreement is not achieved, a *violation* is recorded. SLAs are widely used by CSPs as a basis for defining the quality of service (QoS) a user can expect from a provider [7]. There is however no straightforward way for an easy online analysis of the expressed QoS. A provider generally relies on its own engineering approaches and experience to guaranty a QoS. Verification or certification usually provides needed analyses to ensure them. As the QoS is intricate to handle online, our work intends to study SLA violation based on monitoring data characterizing the behavior of system level resources that we can obtain from commodity servers or VMs at runtime. Monitoring data are system-level metrics that are easy to obtain from commodity servers of VMs at runtime.

Thus, this paper proposes an approach (we call it *detection approach*) to detect SLA violations, by means of machine learning *models* and based on monitoring data. The detection of SLA violations can lead a system administrator to reboot an anomalous VM or horizontally scale a VNF (i.e., add more nodes). In addition to this detection, it would be of interest to detect preliminary symptoms of such violations, or in other terms, to identify the behaviors of a VNF before it leads to an SLA violation. This would enable to adopt proactive measures before a violation actually happens and may disrupt a user experience. Also, when either an SLA violation is imminent or already occurring, it is not enough for administrators to get a global alarm over a large distributed VNF. The root cause of such events, or at least an indication of it, is required to plan adequate measures. In that concern, this paper focuses on these *three objectives* to be tackled online:

- Detection of SLA violations
- Detection of preliminary symptoms of SLA violations
- Detection of SLA violations with root cause localization

The detection of preliminary symptoms of SLA violations should be performed together with the detection of SLA violations. They are complementary objectives. Any time the detection of preliminary symptoms failed, the detection of SLA

violations still can detect the violation as it occurs. Our third objective aims to improve the detection of SLA violations as it also localizes the root cause of the violations. Localization is carried out by either pinpointing a workload being too heavy or identifying anomalous VNF VMs.

Our approach to achieve the three detection objectives is based on models that are designed to perform online. Nevertheless, an offline training phase is needed, during which these models are defined and trained with monitoring data to identify the anomalous VNFs behaviors that we intend to detect. Since a system is mostly exhibiting a normal behavior during runtime, fault injection techniques are used in our approach so as to accelerate the occurrence of behaviors leading to SLA violations. These techniques aim at deliberately provoking anomalous behaviors in a system. They are used in this work in order to build complete monitoring datasets so as to train models. Injections are performed at a local level by means of local injections in single VMs or, at a global level, by submitting heavy workloads.

Two types of analysis of the validation results are also carried out, *per-VM* or globally using an *ensemble analysis* combining per-VM analyses. Moreover, two application-agnostic monitoring sources are studied at the VM level and at the hypervisor level, both for the monitoring of the VNF VMs. Validation experiments are performed on a real case study, namely a virtual IP Multimedia Subsystem (IMS) [8] implemented by the Clearwater project. This IMS implements multiple components, dynamically chained together, each one hosted on a VM.

In the following, we first present our global anomaly detection approach in Section II. The implementation of an operational system based on this approach is presented in Section III. Then we describe our case study and platform in Section IV. Section V provides validation results of our approach. Section VI discusses related work. Finally, Section VII concludes and provide some directions for future work.

II. Global approach

In the following, the three main points composing our global approach are presented. We describe the detection approach and the monitoring data used to detect anomalies in a VNF. Then, we present our fault injection approach aimed at gathering datasets representing several behaviors of a VNF. During an offline training phase, these datasets are used to define models used for online detection.

A. Detection approach

The VNF QoS that an SLA encompasses may be defined by several complementary high level characteristics such as service availability or speed [7]. These characteristics can be described by lower level metrics such as the service response time or the service throughput. As a consequence, we hereby provide some definitions related to the VNFs service level as studied in our work.

The *service level* is evaluated as the percentage of successful completion of user requests by the VNF, in other terms, the percentage of successful requests (PSR). In our work, we evaluate the percentage of unsuccessful requests ($PUR = 1 - PSR$) to study VNF SLA violations. The SLA is satisfied as long as the PUR does not overpass a maximal PUR value called *PUR_max*, or the PSR is not below a minimum value. An *SLA violation* (*SLAV*) happens when PUR_max is overpassed ($PUR > PUR\_max$). A *preliminary symptom of SLAV* is represented by the state describing the system behavior during a period $[t - \delta_t, t]$ where t is the time of occurrence of the SLAV. The state is described by the monitoring data collected during this period.

As stated earlier, three detection objectives are considered i) the detection of SLAV, ii) the detection of SLAV-PS due to heavy workload, and iii) detection of SLAV with root cause localization.

The detection of SLAV or their preliminary symptoms leverages the classification of behaviors into normal and anomalous behaviors. It can pinpoint whether the VNF exhibits an anomalous behavior. The detection with root cause localization is the preeminent detection investigated in this work. It performs detection with online localization of an anomalous behavior cause. In this work, the cause of an anomalous behavior can be an anomalous VM suffering from local performance stressing, or it can also result from a global heavy workload toward the VNF.

The detection of SLAVs or their preliminary symptoms turns out to be a classification problem: is there an SLAV (resp. a preliminary symptom) or not? We even can go further, and in case of violation, can we identify its cause? Machine learning is a famous field of computer science aimed at getting automatic computing procedures to learn a task without being explicitly programmed. It turned out to be extremely relevant for classification problems [9]. Machine learning can be applied to our problem so as to classify behaviors corresponding or leading to SLAVs, or not. As there is an SLAV or a preliminary symptom of SLAV, we consider that a system exhibits an *anomalous behavior*. Otherwise it has a *normal behavior*.

There are a plethora of machine learning algorithms and a lot of them are specifically dedicated to handle numerical data. Actually, monitoring data enable us to directly observe a system and represent its behaviors by means of numerical data. Representing a behavior needs some heavy preprocessing when using audit data, OS logs, or application logs. Besides, this work particularly focuses on the identification of anomalous behaviors from monitoring data of VMs OSs such as CPU consumption, disk I/O, and free memory. As seen in previous researches [10]–[16], these data are well suited to reflect the behavior of a computing system.

With respect to the machine learning *models* that we aim to build for detection (in our case, classifiers), samples of labeled monitoring data are needed to train them to discern different VNF behaviors. Considering our two first objectives, samples should be labeled with two classes corresponding to normal or anomalous behavior so as to train models to classify these two behaviors. As for the third objective, samples should be labeled with several classes according to the several SLA violation origins in the VNF. This method is called *supervised learning* (by opposition to unsupervised learning, i.e., clustering). Samples are called *training data*. They are obtained during a training-purposed runtime phase (i.e., *training phase*) during

which a testbed or a development infrastructure is monitored while experiencing different types of behaviors. Based on our field knowledge about system anomalies that may lead to SLAVs, we can emulate them during a training phase and train models from the collected data. The emulation is performed through fault injection campaigns (they are described in II-C). Once a model is trained from a training dataset, it can be used *online* during a *detection phase*. Thereupon, it performs *predictions* of whether a given monitoring sample belongs to a particular *class* of behaviors that it learned to discern. The ability of models to perform accurate predictions is referred to as *predictive accuracy* in this paper.

Anomaly detection is performed over the monitoring data subdivided into data related to each VM hosting a VNF component. We call it a *per-VM* analysis, as in [10]. By not combining all monitoring data from all VMs, this analysis enables not to accumulate irrelevant features (i.e., monitoring metrics) in one dataset and thus, to improve the predictive accuracy of models. It also eases the comparison of the predictive accuracy of detection models based on different monitoring sources. Finally, it is a possible means to analyze error propagation [17] between VMs through monitoring data.

The predictive accuracy obtained from the combination of the per-VM predictions is also evaluated in this work. We call it an *ensemble analysis*. The ensemble analysis is a means to provide CSP administrators with one single prediction output while giving further access to the per-VM analysis when it is required. Besides, we believe that a study could be performed to identify which types of VNF component (proxy, database, etc.) are best to detect SLAVs caused by particular anomalies. From this study, we could give higher weight to the predictions of VM models that are more likely to give good individual predictive accuracy.

Table I summarizes all acronyms and abbreviations used in this paper.

TABLE I. ACRONYMS AND ABBREVIATIONS TABLE

CSP	**C**ommunication **S**ervice **P**rovider.
SLA	**S**ervice **L**evel **A**greement.
SLAV	**SLA V**iolation.
SLAV-PS	**P**reliminary **S**ymptoms of SLAV.
PSR	**P**ercentage of **S**uccessful **R**equests.
PUR	**P**ercentage of **U**nsuccessful **R**equests. It is the service level characteristic being studied in this work.
PUR_max ...	**Max**imum value of the **PUR** agreed with customers. It stands at the limit between the agreement and the violation. A violation takes place if $PUR > PUR\_max$.
VNF	**V**irtual **N**etwork **F**unction.

B. Monitoring

Monitoring provides units of information about a system that are called performance counters (referred to as *counters*). The actual counter values being collected from a system are called performance metrics (referred to as *metrics*). A vector of metrics collected at a given timestamp corresponds to a monitoring observation (also referred to as *observation*).

In this work, observations related to a VNF VMs hosted in a virtualized infrastructure are collected from *two monitoring sources*, namely the hypervisor, or the OS of the VMs.

- The hypervisor hosting the VNF VMs can provide monitoring data related to each VM it hosts such as the memory, the CPU speed or the network bandwidth that the hypervisor grants to the VM. Such a monitoring source is called *black-box* as it does not need any tool to be installed in the VMs.
- Monitoring data can also be collected directly from the OS of the VNF VMs. Collected observations from VMs require the installation of monitoring agents in VMs. This is referred to as a *grey-box* monitoring source. Considering this source, the number of available counters is more important than in the case of the black-box source since they can relate to the OS performance in terms of system buffers size and use, and in terms of memory pages state for instance. These low level VMs counters cannot be known by the underlying hypervisor.

Both monitoring sources provide periodically observations related to each single VM of the VNF. We evaluate the benefits in terms of the predictive accuracy a grey-box monitoring may provide for the detection objectives. For both monitoring sources, our approach does not require knowledge about the VNF implementation. Therefore, it can easily be applied to all types of VNFs.

C. Fault injection approach

To accelerate the occurrence of faults during the training phase of our detection approach we use fault injection techniques. We particularly focus on injecting anomalous behaviors in the VNF. Our injections emulate widespread anomalies existing in common computing system due to punctual and abrupt behaviors changes (we do not study long term degradations). Such anomalies arise when software or hardware faults are activated.

Two methods are used in order to trigger these anomalies:

- Local emulation of anomalies within VNF VMs (increase of CPU consumption, memory leaks, etc.).
- Global configuration of the benchmark with a heavy workload. In other words, the VNF is stressed at a heavy load beyond which a correct service is not guaranteed and SLAVs are likely to be observed (the load level is set with regard to the platform deployment and its maximum capacities).

Anomaly injections in the VNF depend on three parameters, namely a type, an intensity, and a target VM. By means of our two methods, six types of anomalies are emulated, namely 1) CPU consumption, 2) misuse of memory (i.e. memory increase), 3) anomalous number of disk accesses (i.e. increase of disk I/O access and synchronizations), 4) network packet loss, 5) network latency, and 6) heavy workload. They are respectively referred to as CPU, memory, disk, packet loss, latency, and heavy workload injections, and described bellow.

CPU consumption. Anomalous CPU consumptions may arise from anomalous programs encountering impossible termination conditions leading to infinite loops, busy waits or deadlocks of competing actions, which are common issues in multiprocessing and distributed systems.

Memory leaks. Anomalous memory usages are common whose allocated chunks of memory are not freed after their use. Accumulations of unfreed memory may lead to memory shortage and system failures. We believe that such cloud-related mechanism as ballooning may also lead to such failures.

Anomalous number of disk access. A high number of disk accesses, or an increase of disk accesses over a short period of time, emulates anomalous disks whose accesses often fail and lead to an increase in disk access retries. It may also result from an anomalous program stuck in an infinite loop of data writing.

Network anomaly. Such anomalies may arise from network interfaces or the interconnection of networks. We emulate packet losses, and latency increases. Packet losses may arise from undersized buffers, wrong routing policies and even firewall misconfigurations. Latency anomalies may originate from queuing or processing delays of packets on gateways.

Heavy workload. They emulate a large number of users consuming the VNF service, too important with regard to the testbed deployment.

Finally, two intensities were selected for the local VM anomalies, namely medium and high. We calibrated them based on prior experimentations on our testbed. Regarding the memory, disk and CPU injections, the intensity values of an anomaly are constrained by the capacity of VMs OSs. In other words, the high intensity injection (resp. medium) is the maximum resource consumption (resp. 50% of resource) allowed by the OS. Considering the remaining types of injections, the high intensity injection (resp. medium) value is set so as to lead to around 99% (resp. around 50%) PUR when applied in at least one VNF VM. Indeed, even if all VMs are configured with the same VM template, the components installed in each VM do not use VM resources in a similar way. As a consequence, high intensity injections in VM A could lead to a maximal PUR whereas the same injection in VM B would lead to a 2% PUR. Heavy workloads are carried out by calibrating the benchmark workload for our testbed capacities and trigger SLAVs.

III. Implementation

This section presents the implementation of an operational system based on our approach. We first describe the detection approach implementation, then the monitoring of the VNF and finally the fault injection approach implementation.

A. Detection approach

The Random Forest algorithm is used in order to classify the VNF behaviors. This algorithm is well known to provide good classification results and to handle a large number of vector features (i.e. observations counters) [18]. Although we use a per-VM approach, the number of OS features (more than twenty) are considered as high-dimensional. Besides, once a model is trained, the prediction overhead is extremely low [19]. Finally, the Random Forest algorithm also enables to get insight about the most important metrics that influence the prediction output [19].

The algorithm is configured with ten decision trees, based on preliminary works that showed good results with as many trees. Besides, trained Random Forest models are configured to predict the probabilities of class membership of an observation. We call them *prediction probabilities*. The *output* of a model is therefore a probability. For instance, a model output for an observation can be a probability of 0.2 for it to be in the class labeled 1. Given the resulting probabilities, it is then easy to set a threshold defining the limit probability from which an observation corresponds to an anomalous behavior.

Finally, the models on which relies our approach can be trained and tested on an external machine to the VNF. It can be run either on the monitoring module machine, or on any separate machine. Thus, our approach does not introduce overhead in the VNF runtime.

B. Monitoring

Monitoring data are collected from two separate *monitoring sources* as described in II-A. For both monitoring sources, observations of each VM are collected every 15sec.

The black-box source heavily relies on our VMware underlying infrastructure (see IV-A). The library Pysphere[1] communicates with the VMware SDK and is used to collect VMs metrics from the hypervisor. For each VM, we collect observations composed of the 152 metrics listed online[2].

The grey-box source is implemented by a Ganglia [20] monitoring agent installed in each Clearwater VM. We configured the agent with python modules so as to collect from each VM 227 metrics listed online[3].

During the training-phase, observations are stored in database so as to train offline our models. During the runtime of our operational system, observations should directly be provided to the models in order for them to perform predictions. Nonetheless, the observations of each VM are studied separately in order to perform anomaly detection in each single VM (per-VM approach).

C. Fault injection approach

1) Single injection: Anomalies presented in the description of our approach (II-C) emulate behaviors that could be activated by software, hardware, or configuration faults. We emulate them by software means.

Injections that consist in a heavy workload are carried out by running a load calibrated so as to lead to SLAVs between 2% and 30% of PUR considering our testbed configuration.

Injections that emulate anomalies in the VNF VMs are carried out by injection agents that stress the VNF components individually. Injection agents are installed in each VM. They are run through an SSH connexion orchestrated by a campaign handler which is a configurable entity hosted on any machine apart from the VNF VMs (in order to maintain experiment isolation). We use the Linux kernels tools *iptables* and *tc* for the injection of network latencies on the POSROUTING chain, and *iptables* on the INPUT and OUTPUT chains for the injection of packet losses. The stress test tool Stress-ng[4] is run for CPU, disk, and memory related injections.

[1] https://pypi.python.org/pypi/pysphere

[2] https://homepages.laas.fr/ csauvana/datasets/pysphere_vm_counters.txt

[3] https://homepages.laas.fr/ csauvana/datasets/ganglia_vm_counters.txt

[4] http://kernel.ubuntu.com/~cking/stress-ng/

2) Campaigns: An *injection campaign* targets a single VM. In each campaign, the benchmark is run for 80min. After 40min an anomaly of 3min is injected. The injection time is calibrated so as to affect several instances of benchmark scenario executions (an execution lasts less than 1sec) and at the same time not to be too short regarding our monitoring period of 15sec. The workload request rate during normal behavior periods is calibrated to lead to less than 1% PUR of the VNF.

Finally, for our experiments, several injection campaigns are run so as to get statically significant large datasets. First, a heavy workload is run, then campaigns are run the following way: each injection of each intensity is performed in each VM. At the end of each injection campaign, the VMs are rebooted.

IV. Case Study and Platform

Our testbed is composed of three main entities detailed below: the cloud infrastructure, the Clearwater VNF running on the infrastructure, and the workload. The workload is used for the simulation of VNF users and for the PUR computation that is needed for validation purposes.

A. Infrastructure

The virtualized platform is a VMware vSphere 5.1 private platform composed of 2 servers Dell Inc. PowerEdge R620 with Intel Xeon CPU E5-2660 2.20GHz and 64GB memory. Each server has a VMFS storage. One hypervisor hosts the workload as well as the monitoring module and the other hosts all Clearwater components VMs. Each VM has 2 CPUs, a 10GB memory, a 10GB disk. VMs are connected through a 100Mbps network.

B. Clearwater VNF

Clearwater is an open source implementation of an IMS for cloud platforms. It provides SIP-based (Session Initiation Protocol) voice and video calling, and messaging applications. It implements key standardized interfaces and functions of an IMS (except a core network) which enable industries to easily deploy, integrate and scale an IMS. Clearwater was initially designed for clouds commodity servers. It is consequently well suited for NFV related studies. It encompasses six software components, namely Bono, Sprout, Homestead, Homer, Ralf, and Ellis (Figure 1).

Bono is the SIP proxy implementing the P-CSCF function (Proxy-Call/Session Control Functions). It handles the users requests and routes them to Sprout. It also performs Network Address Translation traversal mechanisms.

Sprout is the IMS SIP router receiving requests from Bono and routing them to the adequate endpoints. It implements some S-CSCF (Serving-CSCF) and I-CSCF (Interrogating-CSCF) functions and gets the required users profiles and authentication data from Homestead. Sprout can also call application servers and actually contains itself a multimedia telephony (MMTel) application server, whose data are stored in Homer (when calls are configured to use its services).

Homestead is an HTTP RESTful server and stores Home Subscriber Server data in a Cassandra database (i.e. information about subscribed services and locations). It is in charge of some I-CSCF and S-CSCF functions.

Thus, Bono, Sprout, and Homestead work together to control the sessions initiated by users and handle the entire CSCF.

Homer is a XDMS (XML Document Management Server) server with an XML Configuration Access Protocol Server (XCAP) interface, and runs a Cassandra database. It stores configuration information about MMTel service.

Ralf is the CTF (Charging Trigger Function). It bills the events collected by Bono and Sprout and reports them to a Charging Data Function server (this server is not included in the Clearwater project).

Ellis is a provisioning portal offering a web interface to users for testing purposes.

The IMS scales-out horizontally by means of a simple DNS load balancing mechanism. Our testbed encompasses Bono, Sprout, Homestead and Homer, each of which is deployed on one VM (see Figure 1). In this study, the billing function is not configured, so Ralf is not included in our testbed. Neither is the testing component Ellis. Also, our deployment does not encompass redundancy. We focus our work on Bono, Sprout and Homestead constituting the entire CSCF: we perform injections and provide evaluation results for these VMs.

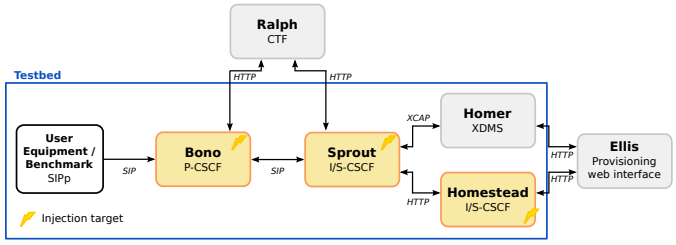


Fig. 1. Testbed deployment and Clearwater components.

C. Workload and PUR evaluation

In order to validate our approach we need a tool to simulate our VNF workload and from which to compute our PUR. IMS workloads for Clearwater can be emulated by means of the SIPp benchmark[5]. The benchmark workload can be configured with a number of calls per second (we call it *load parameter*) to be sent to the IMS. A call corresponds to the execution of a *scenario* . A scenario is described in terms of SIP transactions in XML and a workload is defined by a scenario and a load parameter. A SIP transaction is a SIP message to be sent and an expected SIP response message. A call fails when a transaction fails. A transaction may fail for two reasons: either a message is not received within a precise duration which triggers a timeout, or an unexpected message is received. Unexpected messages are the HTTP error codes 500 (Internal Server Error), 503 (Service Unavailable) and 403 (Forbidden). Also, in our testbed, the PUR more exactly accounts for the percentage of unsuccessful calls.

Now considering our validation experiments, the selected scenario simulates a standard call between two users and encompasses the standard SIP REGISTER, INVITE, UPDATE, and BYE messages (additional details about the load during

[5]http://sipp.sourceforge.net/index.html

experiments are presented in II-C). The SLA of the VNF corresponds to the PUR computed for all users calls simulated by the benchmark. (for instance a real case context would be: a main company asked for the use of a VNF for several of its own users and agreed on a PUR value). The benchmark simulates all users and by the analysis of its logs we can compute the combined PUR for all users. Finally, a SLAV-PS is regarded as the behavior the VNF VMs observe over a time window before an SLAV happens.

V. Validation results

In order to validate our approach, we collected one dataset for each monitoring source from our testbed so as to train models and to make them perform predictions. In the following, we first present the preparatory work performed on the monitoring observations before running the training phase of our models. The validation metrics used to evaluate the predictive accuracy of the models are presented in a second part. Then, our validation results for the detection phase are presented. We present results related to the detection of SLAV and the detection with root cause localization of anomalous VMs. Then we show results related to the detection of SLAV-PSs.

Detection results are presented with a per-VM and an ensemble analysis and also depending on the monitoring source namely black-box or grey-box (hypervisor or monitoring agent). Considering the ensemble analysis, we take the mean prediction probabilities of each model related to the detection of the same event across all VMs.

A. Preparatory work

In our experimental validation, we focus on Bono, Sprout, and Homestead VM observations corresponding to the entire CSCF. Injections are performed in these three VMs and results are provided considering the observations of these VMs. When analyzing a model trained only with VM A observations, we call VM A the *detection-VM*. When similar results are observed from models trained with our three different detection-VMs, only Bono results are provided.

In order to perform validation, for each monitoring source, a dataset representing several behaviors of our VNFs is assembled. It is large enough to train detection models and to test them. For both monitoring sources, around 16500 observations for each VM are collected while we ran the injection campaigns presented in II-C (global dataset of around 66000 observations). Before the creation of models, our validation dataset is shuffled and split into 60% of training data and 40% of testing data from which our results are computed.

Moreover, K-fold cross validation has been performed on the VMs observations associated with 5 classes (normal behavior, heavy workload, injections in Bono, injections in sprout, injections in homestead). For a 10-fold case, a small standard deviation of 0.2 was observed for both precision and recall metrics which means that the following results are relevant and easily generally applicable.

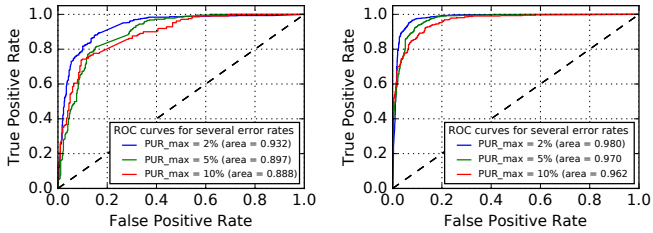
B. Validation metrics of the operational system

The validation of the machine learning models to classify anomalous behaviors is based on several metrics, namely the Receiver Operating Characteristic (ROC) curve, as well as the precision, recall and F1-score. ROC curves are obtained by computing performance measures for different thresholds of the *prediction probabilities* (see III-A for more details). The ROC curve corresponds to the true positive rate (TPR) against the false positive rate (FPR). A perfect classifier would have an area under the curve (AUC) of 1.

The ROC curves are relevant to study the predictive accuracy of our models and their consistency while the prediction probability thresholds change. They enable us to evaluate our approach with metrics that do not change with the proportion of normal or anomalous behaviors in the validation dataset. In other words, they are not sensitive to the class skew and the results can easily be generalized to several case scenarios of detection with different baseline distributions. Also, the FPR (i.e., the rate of false alarm) can easily be analyzed and it is of interest in our domain. Indeed, it is expensive (in terms of time and money) for a CSP to take counter measures on components, and the fewer false alarms there are, the better is the approach. As for precision and recall, they are defined by the well known formulas: $Precsion = \frac{TP}{TP+FP}$ and $Recall = \frac{TP}{TP+FN}$ where TP are the true positives, FP are the false positives, and FN are the false negatives. The F1-score corresponds to the harmonic mean between precision and recall. These metrics provide a precise evaluation of the predictive accuracy, that can give an account of the efficiency of our method for a CSP. They are defined with a given prediction probability threshold.

C. Detection of SLAV

1) Per-VM analysis: The aim is to analyze the ability of our approach to detect SLAVs on a per-VM basis. In this analysis, we also test whether models trained to detect SLAs defined with different values of PUR_max have different predictive accuracies. Three models were actually trained to detect violations with PUR_max = $\{2\%, 5\%, 10\%\}$. By this mean we can quickly notice whether our models are sensitive to PUR_max.



(a) Bono, black-box (b) Bono, grey-box

Fig. 2. ROC curves for SLAV detection: Bono observations, detection only, with PUR_max = {2%, 5%, 10%}.

Figure 2-a presents the ROC curves of three models trained with Bono observations to detect the three values of PUR_max, for a black-box monitoring. The curve related to PUR_max = 2% has an AUC of 0.93, compared to 0.89 for PUR_max = 10%. This shows that the higher is PUR_max (i.e., the lower is the SLA expectation), the lower is the TPR. We believe the reason for this lower predictive accuracy is that some injections change the VMs behaviors enough for the model to detect the change but not enough for an SLAV to be well identified. For

all values of PUR_max, the ROC curves reach at least a 0.80 TPR with a 0.2 FPR. Results are similar when models are trained from the observations of Sprout and Homestead.

Also, Table II shows precision, recall and F1-score for the detection depending on the detection-VM and PUR_max values. From all detection-VMs, SLAVs can be detected with good precision an recall with a minimum of 0.97 precision and 0.90 recall for PUR_max = 10%. The best precision and recall are always obtained with Sprout observations. Sprout has indeed a central role in the Clearwater CSCF. This means that the black-box monitoring can be used by CSPs as a first step toward the triggering of reconfiguration actions with very good predictive accuracy.

TABLE II. PREDICTIONS OF SLAV: DETECTION ONLY, GREY-BOX.

Monitoring observation	Measure	PUR_max= 2%	PUR_max= 5%	PUR_max= 10%
Bono	Precision	0.92	0.91	0.88
	Recall	0.95	0.95	0.90
	F1-score	0.93	0.93	0.89
Sprout	Precision	0.92	0.90	0.90
	Recall	0.95	0.94	0.93
	F1-score	0.93	0.92	0.91
Homestead	Precision	0.89	0.88	0.87
	Recall	0.93	0.90	0.91
	F1-score	0.90	0.89	0.89
Ensemble analysis	Precision	0.92	0.90	0.89
	Recall	0.96	0.95	0.93
	F1-score	0.94	0.93	0.91

Figure 2-b presents the ROC curves of the models trained with the same PUR_max value as for Figure 2-a and with a grey-box monitoring. The ROC curves for the different PUR_max are more significantly close to each other than in Figure 2-a but still, the case where PUR_max = 2% leads to a better predictive accuracy with a ROC AUC of 0.98. Moreover, ROC results are significantly better compared to black-box monitoring.

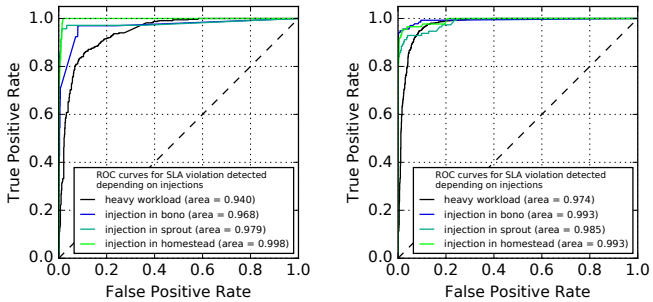
This last remark is notably due to the larger variety of metrics that can be retrieved from a monitoring agent. An other reason is that these metrics are updated more frequently. Indeed, we regularly noticed in our dataset that some observations collected from the hypervisor are not updated at strict regular intervals as expected from the configuration. In other words, successive observations from a dataset can happen to be the same although they are collected 15s apart (i.e., the period of data collection). Thus, we believe that with a reasonably sound and higher data collection period, we could achieve better results with the black-box monitoring.

2) Grey-box ensemble analysis: Table V (last row) presents the precision, recall and F1-score for the combined prediction probabilities of all detection-VMs. The F1-score has a mean value of 0.93 across all PUR_max values. Its lower value is 0.91 for PUR_max = 10% and it reaches 0.94 with PUR_max = 2%.

D. Detection with root cause localization

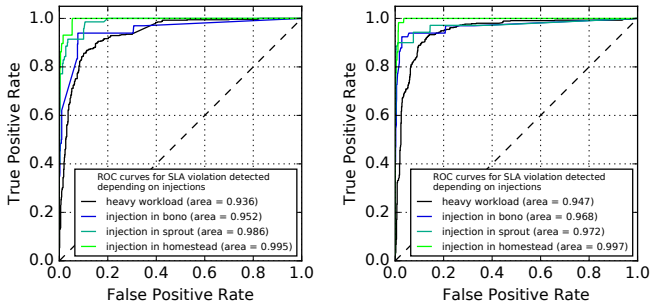
1) Root cause localization in anomalous VMs: For each detection-VM, four models are trained to classify SLAVs caused by a heavy workload, injections in Sprout, Bono and Homestead respectively. The PUR_max is set to 2%.

Figures 3-a, 4-a, and 4-b present the ROC curves of the detection with root cause localization of models trained respectively on Bono, Sprout and Homestead observations with a black-box monitoring. It can be noticed that SLAVs due to injections in Homestead lead to the lower FPR compared to injection in Bono and Sprout. Indeed it achieves a ROC AUC mean over the three VMs of almost 1. This SLAV cause is well detected by models trained from Bono, Sprout and Homestead observations. The heavy workload is detected with the lower TPR. However, it is detectable over all detection-VMs with a mean ROC AUC of 0.94.



(a) Bono, black-box (b) Bono, grey-box

Fig. 3. ROC curves for SLAV detection: Bono observations, detection with root cause localization.



(a) Sprout, black-box (b) Homestead, black box

Fig. 4. ROC curves for SLAV detection: Sprout (a) and Homestead (b) observations, detection with root cause localization, black-box.

TABLE III. PREDICTIONS OF SLAV: BONO OBSERVATIONS, DETECTION WITH ROOT CAUSE LOCALIZATION, BLACK-BOX.

Monitoring observation	Measure	Cause in Bono	Cause in Sprout	Cause in Homestead	Cause by heavy load
Bono	Precision	0.87	1.00	1.00	0.86
	Recall	0.63	0.57	0.57	0.85
	F1-score	0.74	0.72	0.72	0.86

Table III presents the precision, recall, and F1-score considering Bono observations and depending on the SLAVs cause, for a black-box monitoring (results from Sprout and Homestead observations follow a similar trend). The F1-score does not overpass 0.86 which is obtained for the detection of high workload SLAVs. Also, the SLAVs originated from Sprout and Homestead are detected with a very low recall of 0.57. In other words, several true alarms are missed. The black-box monitoring is in this case of detection with root cause localization not sound enough to be applied by a CSP.

Figure 3-b presents the ROC curves of the detection with root cause localization of models trained from Bono observations with a grey-box monitoring. Here again, the predictive accuracy is better than with the black-box monitoring. Indeed, the mean ROC AUCs for all causes is close to 1 whereas it was of 0.97 with black-box monitoring. The heavy workload is in this case not detected with a TPR significantly below to the TPR of other causes. Indeed, the corresponding ROC curve shows that there exists a probability threshold that enables a TPR of 0.94, and a FPR of 0.05, which is an excellent predictive accuracy that can convince a CSP to use our approach. Thus, there is a higher FPR on average in black-box monitoring, except for the detection of SLAVs originated from Homestead which has in both monitoring sources a ROC AUC of almost 1. A grey-box monitoring is therefore more suitable when it is possibly deployable. This is consequently due to the lack of some relevant observation counters in the black-box monitoring data. The Random forest implementation actually provides a ranking of the most important counter enabling correct detection and we indeed notice that the most influent counters are counters linked to the TCP/IP stack that are not known from the basic hypervisor monitoring (for instance the TCP forward retransmission, or the TCP fast retransmission failure).

Table IV presents the precision, recall and F1-score obtained for all detection-VMs depending on the SLAV cause and in grey-box monitoring. This table enables us to get more insight about the predictive accuracy of our models in general for a real case scenario. SLAVs caused by a heavy workload are detected with the highest recall but not the highest precision due to its low TPR shown in previous figures of this subsection. We notice that Homestead observations enable the detection of SLAVs originating from Sprout (precision and recall of 0.99 and 0.85). More generally speaking, SLAVs originating from a VM A can be detected from observations of VM B. Moreover, across all VMs, SLAV originated from a VM A can be detected with at least 0.90 precision and 0.80 recall. We can conclude that our approach enables to detect SLAVs and localize their cause while training models either on the observations of Bono, Sprout or Homestead.

TABLE IV. PREDICTIONS OF SLAV: ALL DETECTION-VMS, DETECTION WITH ROOT CAUSE LOCALIZATION, GREY-BOX.

Monitoring observation	Measure	Cause in Bono	Cause in Sprout	Cause in Homestead	Cause by heavy load
Bono	Precision	0.98	0.99	1.00	0.93
	Recall	0.86	0.87	0.83	0.94
	F1-score	0.92	0.92	0.91	0.93
Sprout	Precision	0.99	0.98	0.98	0.92
	Recall	0.80	0.90	0.80	0.93
	F1-score	0.89	0.94	0.88	0.92
Homestead	Precision	0.99	0.99	0.96	0.90
	Recall	0.79	0.85	0.83	0.94
	F1-score	0.88	0.92	0.89	0.92
Ensemble analysis	Precision	0.99	0.98	0.99	0.92
	Recall	0.82	0.93	0.83	0.96
	F1-score	0.90	0.95	0.89	0.94

Moreover, since SLAVs originating from a VM A can be detected from observations of VM B, we conclude that an *error propagation* [17] exists between the VNF components and that we can identify it in monitoring observations. We also show that the observations of a VM A are in general the best observations to identify an SLAV caused by A. This result is relevant since our service level is defined by the PSR which is of a different level compared to local level performance metrics. An hypothesis could have been that the router VM is in any case the VM performing the best prediction given its central role.

Another remark can be derived from the comparison of Figures 2 and 3. It can be noticed that the detection of SLAVs related to either injections in Bono, Sprout or Homestead (Figure 3, injections in VMs curves) leads to ROC AUCs that are closer to 1 than the detection of SLAVs grouping SLAs originated from all types of events (Figure 2, PUR_max = 2%). This means that the dissociation of classes of SLAVs enables us to obtain better predictive accuracy. As a consequence, the separation of SLAVs according to their cause facilitates the decision trees of the models to get more insight about the violations and to make more clear-cut decision conditions. Notwithstanding, this last point should be analyzed with caution. In the case a VNF component changes very frequently its behavior or the workload changes abruptly, the predictive accuracy might not be as good since the decisions conditions are too tightly adapted to a few precise behaviors.

2) Grey-box ensemble analysis: Table IV (last row) presents the predictive accuracy obtained while combining the prediction probabilities of our three VMs, in grey-box monitoring, for a detection with root cause localization of anomalous VMs. We compare these results to the predictive accuracy obtained from Bono observations in grey-box monitoring. Results show that the predictive accuracy for the combination of probabilities leads to similar results as the per-VM analysis in the case of SLAVs originated from Homestead. SLAVs originated from Bono are detected with more precision (0.99) but less recall (0.82). Finally, SLAVs originated from Sprout and heavy workload are detected with more precision and recall.

Thus, an analysis combining prediction probabilities eases the use of anomaly detection for a VNF composed of several components. It also provides similar results compared to the results of individual VMs. It is consequently a good analysis of anomaly detection outputs in VNFs with root cause localization.

E. Detection of SLAV-PS

Given that SLA violations due to a heavy workload are not caused by injection agents run in VMs, but by faults activated by the load, we can create models that are trained for the detection of SLA violation preliminary symptoms due to heavy workloads. We hereby study models that are trained for the detection of SLAV-PSs. Preliminary symptoms are computed over a time window of $75s$ before an SLAV happens (it actually corresponds to the five monitoring observations before the violation considering a $15s$ monitoring period, see III-B). Results with a black-box monitoring are not presented as they do not achieve acceptable precision and recall for CSPs to use it (the F1-score is most of the time less than 0.70 with a detection with root cause localization).

1) Grey-box per-VM analysis: Table V presents the precision, recall and F1-score for the three detection-VMs and depending on the PUR_max value. It shows that the best predictive accuracy is obtained with PUR_max = 5% for any

detection-VM, and it has a F1-score average across all VMs of 0.89. These results show that our approach for detection only of SLAV-PSs is efficient and could be applied by a CSP. Results are similar independently from the PUR_max value.

TABLE V. PREDICTIONS OF SLAV-PS: BONO OBSERVATIONS, DETECTION ONLY, GREY-BOX.

Monitoring observation	Measure	PUR_max= 2%	PUR_max= 5%	PUR_max= 10%
Bono	Precision	0.79	0.86	0.80
	Recall	0.89	0.93	0.93
	F1-score	0.84	0.89	0.86
Sprout	Precision	0.79	0.85	0.83
	Recall	0.92	0.94	0.94
	F1-score	0.85	0.89	0.88
Homestead	Precision	0.80	0.84	0.82
	Recall	0.91	0.92	0.93
	F1-score	0.85	0.88	0.87
Ensemble analysis	Precision	0.80	0.85	0.82
	Recall	0.95	0.95	0.96
	F1-score	0.75	0.90	0.89

2) Grey-box ensemble analysis: Table V (last row) presents the precision, recall and F1-score for the combined prediction probabilities of all detection-VMs. The F1-score has a value of 0.75 with PUR_max = 2% and even reaches 0.90 with the PUR_max = 10%. This result is quite promising and could be considered for the use by a CSP.

F. Results summary

Validation results show that the detection of SLAV-PSs due to a heavy workload with grey-box monitoring is efficient and could be applied by a CSP. Good results are obtained when working on a per-VM or an ensemble analysis. The detection of SLAVs with black-box monitoring, can be used by a CSP to detect our anomalies with a mean precision and recall of 0.91 and 0.94 over different PUR_max values (PUR_max $\in \{2\%, 5\%, 10\%\}$). Best prediction results are obtained with the Sprout component which has a central role in Clearwater. We would argue that it should be tested whether detection tends to be generally better from major components of a VNF. ROC, precision and recall results are better in grey-box monitoring than in black-box monitoring. It is the case for all validation results of our objectives.

In detection with root cause localization, we show that SLAVs originating from a VM A can be detected from observations of VM B. There is indeed an error propagation that enables us to locate anomalous VMs with monitoring data with high predictive accuracy. An ensemble analysis combining all per-VM predictive probabilities can also be applied while keeping high predictive accuracy in grey-box monitoring. The ensemble analysis provides one single output of detection and it consequently eases the proactive analysis of detection alarms. Thus, it is a good for detection of SLAVs with root cause location.

As a result, a model for a detection only of SLAV-PSs as well as models for the detection with root cause localization of SLAVs should be used together. The first detection model enables to warn CSP administrators to take appropriate proactive reconfiguration decisions (it would depend on the SLA of the customers). When the models for the detection of SLAVs raise alerts, the CSP administrator should configure its platform to automatically and directly reconfigure its VNF. The higher is a configured customer SLA, the more carefully the CSP administrator should examine the alarm caused by SLAV-PSs.

Considering the application of our anomaly detection approach in a real case deployment, several points are to be addressed. The training phase of our approach should be performed offline or on a development infrastructure when it is possible. The training duration depends on the number of different VNF components. Models can be replicated when used for the detection in VMs hosting the same VNF component and with the same resources. It is actually widespread to use templates of VMs and work with VMs with identical resources. Since resource consumption of VMs in real deployments is predictable [21], resource reconfiguration like hot plugs of CPU or memory are rare. This largely limits the online reconfiguration of the VMs resources, and motivates the use of VMs templates. As a new VNF VM is deployed, no new training dataset is needed in the case the new VM follows the same template as previously deployed VMs. Notwithstanding, models of our approach should at times (depending on the VNF environment reconfiguration period) be trained offline from new datasets comprising new anomalous behaviors tightly linked to new faults that can occur in the VMs with their evolving normal behaviors. Finally, for a real case scenario, the detection phase operating online should be handled by a big data processing framework (such as Apache Spark) in order to quickly manage several models for several VMs.

VI. RELATED WORK

NFVs and related paradigms aimed at introducing innovation inside networks like Software-Defined Networking have been heavily investigated [22], [23]. However, bringing networks in automated software-based platforms leads to all the more intricate SLA violation scenarios. Recent research results show the sensitivity of NFV to IT failures. In [24], Ge et al. show that NFV on commodity servers may need hardware acceleration to handle network processing tasks such as Deep Packet Inspection, Network Deduplication, or Network Address Translation. Moreover, some virtualization features like live VM migration may also represent a source of performance degradation for VNFs since it may cause unacceptable downtime [25]. In order to counter these issues, most of recent research work related to VNF is focused on the implementation of VNF dedicated OS as ClickOS [26] a virtualized platform optimized for middlebox processing, or the implementation of a single VNF like virtualized Deep Packet Inspection functions (vDPI), offered as a cloud service [27] or Software Defined Monitoring functions by Choi et al. [28].

Our work deals with anomaly detection in VNFs in particular which is a field not thoroughly explored. However, it has been extensively studied for cloud infrastructures. In previous work [19] we defined Tejo, an approach based on machine learning for anomaly detection in databases. In this paper we evaluate such an approach for VNF while tackling error propagation in VMs as well as preliminary symptoms of SLA violations. DAPA performance diagnostic framework [29] models the existing relationships between application performance and underlying system metrics to detect SLA

violations. The SLA violation diagnostic is performed afterward with an unsupervised learning algorithm. PeerWatch [30], an anomaly detection and diagnosis tool, uses a correlation analysis to model the relationship existing between the components of distributed applications, and detects an anomaly when the correlation between these components drops significantly. In comparison to these works, our approach proposes to detect *online* causes of SLA violations. FChain [11] monitors the running of distributed applications to detect performance anomalies and to pinpoint the faulty component by reconstructing the propagation patterns of abnormal change points. Our approach has the advantage not to depend on strict synchronisation constraints between several components and the master diagnosis entity. The reconstruction is based on which component started to fail in a first place. In [10] the authors present PREPARE which addresses the problem of anomaly prevention in virtualized cloud infrastructures on a per-VM basis by means of supervised learning methods. In addition, our work is specially intended to detect SLA violations and their preliminary symptoms, and not only to detect potentially anomalous VMs. Indeed, potentially anomalous VMs may not lead to SLA violation which is actually the problem of interest for CSP. We go further pinpointing anomalous VMs and evaluate to what extent they lead to global service impact.

Unlike these contributions, our approach only relies on monitoring data and no other specification of the VNFs is required. We also evaluate it on the opensource Clearwater project IMS and show that it is capable to detect preliminary symptoms of SLA violations.We also perform online root cause localization.

VII. Conclusions and Future work

In this practical experience report, we evaluated the efficiency of a new anomaly detection approach to cope with three detection objectives, namely the detection of SLA violations, the detection of preliminary symptoms of SLA violations, and the detection with root cause localization. Our approach is based on the Random Forest algorithm and application-agnostic VNF VMs monitoring data. We validated it on a Clearwater testbed, with black-box and grey-box monitoring sources. A per-VM and an ensemble analysis were validated. The ensemble analysis is a means to provide CSP administrators with one single prediction output while giving further access to the per-VM analysis when it is required during root cause analysis. Results show that we can detect our first objective with high predictive accuracy using black-box source. Our three objectives can be detected with excellent predictive accuracy using the grey-box source. Grey-box monitoring always lead to better results compared to black-box monitoring because it encompasses low level metrics notably TPC/IP metrics. Grey-box monitoring is notably essential for the detection of preliminary symptoms of SLA violations and for the root cause localization. We also show that error propagation enables to efficiently localize root causes of SLA violations in each VNF VM. Moreover, results show that an ensemble approach combining all per-VM prediction outputs can ease the proactive analysis of detection alarms and keep high predictive accuracy for our three detection objectives.

In this work, a first validation of our approach was performed by means of Clearwater. We aim for future work to study a more representative case study with load balancing and redundancy. As for the fault injection process, we intend to expose our platform to cloud management actions such as ballooning and live VM migration. In this way we will test whether and to what extent such actions might impact the IMS operation and whether the NFV management and orchestration layer should make use of such techniques.

References

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

[2] W. Lee, S. Stolfo, and K. Mok, "A data mining framework for building intrusion detection models," in *Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on*, 1999, pp. 120–132.

[3] L. Heberlein, "Network security monitor (nsm)–final report." 1995.

[4] J. Zhang, M. Zulkernine, and A. Haque, "Random-forests-based network intrusion detection systems," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 38, no. 5, pp. 649–659, Sept 2008.

[5] E. Aleskerov, B. Freisleben, and B. Rao, "Cardwatch: a neural network based database mining system for credit card fraud detection," in *Computational Intelligence for Financial Engineering (CIFEr), 1997., Proceedings of the IEEE/IAFE 1997*, Mar 1997, pp. 220–226.

[6] W. Lee and D. Xiang, "Information-theoretic measures for anomaly detection," in *Security and Privacy, 2001. S P 2001. Proceedings. 2001 IEEE Symposium on*, 2001, pp. 130–143.

[7] ETSI Technical Report 103 125 V1.1.1. (2012) Cloud; slas for cloud services. http://www.etsi.org/deliver/etsi_tr/103100_103199/103125/01.01.01_60/tr_103125v010101p.pdf.

[8] Project Clearwater. (2016) http://www.projectclearwater.org/.

[9] D. Michie, D. J. Spiegelhalter, C. C. Taylor, and J. Campbell, Eds., *Machine Learning, Neural and Statistical Classification*. Upper Saddle River, NJ, USA: Ellis Horwood, 1994.

[10] Y. Tan, H. Nguyen, Z. Shen, X. Gu, C. Venkatramani, and D. Rajan, "Prepare: Predictive performance anomaly prevention for virtualized cloud systems," in *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on*, 2012, pp. 285–294.

[11] H. Nguyen, Z. Shen, Y. Tan, and X. Gu, "Fchain: Toward black-box online fault localization for cloud systems," in *Distributed Computing Systems (ICDCS), 2013 IEEE 33rd International Conference on*, July 2013, pp. 21–30.

[12] Z. Gong, X. Gu, and J. Wilkes, "Press: Predictive elastic resource scaling for cloud systems," in *2010 International Conference on Network and Service Management*, Oct 2010, pp. 9–16.

[13] D. J. Dean, H. Nguyen, and X. Gu, "Ubl: Unsupervised behavior learning for predicting performance anomalies in virtualized cloud systems," in *Proceedings of the 9th International Conference on Autonomic Computing*, ser. ICAC '12. New York, NY, USA: ACM, 2012, pp. 191–200.

[14] Q. Guan and S. Fu, "Adaptive anomaly identification by exploring metric subspace in cloud computing infrastructures," in *2013 IEEE 32nd International Symposium on Reliable Distributed Systems*, Sept 2013, pp. 205–214.

[15] G. Silvestre, C. Sauvanaud, M. Kaâniche, and K. Kanoun, "An anomaly detection approach for scale-out storage systems," in *26th International Symposium on Computer Architecture and High Performance Computing*, Paris, France, Oct. 2014.

[16] C. Sauvanaud, G. Silvestre, M. Kaâniche, and K. Kanoun, "Data Stream Clustering for Online Anomaly Detection in Cloud Applications," in *11th European Dependable Computing Conference (EDCC 2015)*, Paris, France, Sep. 2015.

[17] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *Dependable and Secure Computing, IEEE Transactions on*, vol. 1, no. 1, pp. 11–33, 2004.

[18] A. Verikas, A. Gelzinis, and M. Bacauskiene, "Mining data with random forests: A survey and results of new tests," *Pattern Recognition*, vol. 44, no. 2, pp. 330 – 349, 2011.

[19] G. Silvestre, C. Sauvanaud, M. Kaâniche, and K. Kanoun, "Tejo: A Supervised Anomaly Detection Scheme for NewSQL Databases," in *7th International Workshop on Software Engineering for Resilient Systems (SERENE 2015)*, Paris, France, Sep. 2015.

[20] M. L. Massie, B. N. Chun, and D. E. Culler, "The ganglia distributed monitoring system: Design, implementation and experience," *Parallel Computing*, vol. 30, p. 2004, 2003.

[21] R. Birke, A. Podzimek, L. Y. Chen, and E. Smirni, "State-of-the-practice in data center virtualization: Toward a better understanding of vm usage," in *2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, June 2013, pp. 1–12.

[22] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *Communications Surveys Tutorials, IEEE*, vol. 18, no. 1, pp. 236–262, Firstquarter 2016.

[23] D. Kreutz, F. M. V. Ramos, P. E. Verssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan 2015.

[24] X. Ge, Y. Liu, D. H. Du, L. Zhang, H. Guan, J. Chen, Y. Zhao, and X. Hu, "Openanfv: Accelerating network function virtualization with a consolidated framework in openstack," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 353–354, Aug. 2014.

[25] Openstack. (2016) Live Migration at HP Public Cloud. https://www.openstack.org/summit/vancouver-2015/summit-videos/presentation/live-migration-at-hp-public-cloud.

[26] J. Martins, M. Ahmed, C. Raiciu, V. Olteanu, M. Honda, R. Bifulco, and F. Huici, "Clickos and the art of network function virtualization," in *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*. Seattle, WA: USENIX Association, Apr. 2014, pp. 459–473.

[27] A. Bremler-Barr, Y. Harchol, D. Hay, and Y. Koral, "Deep packet inspection as a service," in *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '14. New York, NY, USA: ACM, 2014, pp. 271–282.

[28] T. Choi, S. Kang, S. Yoon, S. Yang, S. Song, and H. Park, "Suvmf: Software-defined unified virtual monitoring function for sdn-based large-scale networks," in *Proceedings of The Ninth International Conference on Future Internet Technologies*, ser. CFI '14. New York, NY, USA: ACM, 2014, pp. 4:1–4:6.

[29] H. Kang, X. Zhu, and J. L. Wong, "Dapa: Diagnosing application performance anomalies for virtualized infrastructures," in *Presented as part of the 2nd USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services*. Berkeley, CA: USENIX, 2012.

[30] H. Kang, H. Chen, and G. Jiang, "Peerwatch: A fault detection and diagnosis tool for virtualized consolidation systems," in *Proceedings of the 7th International Conference on Autonomic Computing*, ser. ICAC '10. New York, NY, USA: ACM, 2010, pp. 119–128.