Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 94 (2016) 491 – 496

Procedia Computer Science

International Workshop on Applications of Software-Defined Networking in Cloud Computing (SDNCC 2016)

A system architecture for real-time anomaly detection in large-scale NFV systems

Anton Gulenko[a], Marcel Wallschläger[a], Florian Schmidt[a], Odej Kao[a], Feng Liu[b] *

[a]*Technische Universität Berlin (TU Berlin), Complex and Distributed IT Systems (CIT), 10587 Berlin, Germany*
[b]*Huawei European Research Center, Huawei Technologies Co., Ltd., 80992 Munich, Germany*

Abstract

Virtualization as a key IT technology has developed to a predominant model in data centers in recent years. The flexibility regarding scaling-out and migration of virtual machines for seamless maintenance has enabled a new level of continuous operation and changed service provisioning significantly. Meanwhile, services from domains striving for highest possible availability – e.g. from the telecommunications domain – are adopting this approach as well and are investing significant efforts into the development of Network Function Virtualization (NFV). However, the availability requirements for such infrastructures are much higher than typical for IT services built upon standard software with off-the-shelf hardware. They require sophisticated methods and mechanisms for fast detection and recovery of failures. This paper presents a set of methods and an implemented prototype for anomaly detection in cloud-based infrastructures with specific focus on the deployment of virtualized network functions. The framework is built upon OpenStack, which is the current de-facto standard of open-source cloud software and aims at increasing the availability and fault tolerance level by providing an extensive monitoring and analysis pipeline able to detect failures or degraded performance in real-time. The indicators for anomalies are created using supervised and non-supervised classification methods and preliminary experimental measurements showed a high percentage of correctly identified anomaly situations. After a successful failure detection, a set of pre-defined countermeasures is activated in order to mask or repair outages or situations with degraded performance.

Peer-review under responsibility of the Conference Program Chairs

Keywords: Anomaly detection; Cloud; OpenStack; Fault tolerance; NFV

* Anton Gulenko. Tel.: +49 (30) 314-25286; fax: +49 (30) 314-21114.
E-mail address: anton.gulenko@tu-berlin.de

Peer-review under responsibility of the Conference Program Chairs
doi:10.1016/j.procs.2016.08.076

1. Introduction

The virtualization as a key IT technology in the last years reached a high level of maturity leading to the implementation of IT services on virtualized platforms as a predominant model in data centers. The flexibility regarding scaling-out and migration of virtual machines for seamless maintenance enabled a new level of continuous operation and changed the service provisioning significantly. Meanwhile, not only standard IT services are deployed on virtualized infrastructures, but also services from domains striving for highest availability possible – such as the telecommunication branch – are adopting this approach as well and are investing significant efforts into the development of standard products such as OpenStack[1] to fit the requirements of modern Network Function Virtualization (NFV) scenarios.

These efforts elevate the virtualization infrastructure to the next level of complexity. Critical services e.g. in telecommunication scenarios require an availability of 99.9999% as a standard, which is expected from virtualized solutions as well. The availability requirements for the infrastructures are even increasing, as telecommunication providers widely deploy NFV setups in the context of e.g. OpenStack and thus run essential parts of the critical infrastructure on typically off-the-shelf components. Such platforms lack availability features engineered over decades by the telecommunication community and are still far away from the desired degree of availability. This paper describes an approach and an implemented framework to enable reliable deployment and operation on unreliable components. The basic approach is well-known from other domains, e.g. implementing secure channels over unsecure public networks or building redundant storage arrays from standard hard disks. Analogously, we aim at developing and deploying methods for extensive and in-depth monitoring of the vital system data for failure detection and activation of fault tolerance mechanisms utilizing redundant components.

The monitoring system collects and analyses on-the-fly data from several hardware components and the core operating and virtualization processes in order to detect anomalies that can lead to overall performance degradation, violating the promised quality of service, and eventually lead to crashes of parts of the system. Detecting and handling anomalies is a non-trivial issue which needs to span over a multitude of layers and components composing an NFV system. Due to their large scale such systems must be designed to automatically detect and handle anomalies.

The developed system framework facilitates real-time anomaly detection in large-scale NFV deployments. Anomalies are detected by performing deep cross-layer data collection and mining the collected data through a variety of data analysis techniques. The results are used to select and execute appropriate restoration routines. The data collection implements a continuous feedback loop notifying the restoration engine about the success of the executed routines. The developed methods and tools are included in a productive OpenStack installation, but can be generalized for different systems for Cloud and resource management.

The remaining of the paper is organized as follows. The next section lists related work on fault tolerance for virtualized environments, in particular related to continuous operation of NVF-based systems. Section 3 describes the global architecture and the functionality of the deployed components. Subsequently, in Section 4 we describe the system prototype and give preliminary performance evaluation results. Finally, Section 5 concludes with an outline of future extensions to the presented system.

2. Related work

The importance of anomaly detection in NVF-based systems running in a Cloud management system is reflected in a wide range of research publications. Gaikwad et al. for example developed an integrated workflow for finding anomalies in scientific workflows and execution of applications in cloud-based infrastructures. The authors used an auto-regression approach based on statistical methods for online monitoring in order to find anomalies in the collected data[1]. F²PM is a framework for a machine learning system that predicts the Remaining Time to Failure (RTTF) of software services[3]. The F²PM system collects time series of system-level features such as CPU, memory and swap, using the Lasso regularization as a feature selection method. The authors compared failure prediction approaches on all parameters using an e-commerce application use case.

Alonso et al. describe the Lasso Regularization on e-commerce environment as well[4] and categorize machine-learning classifiers detecting dynamic and non-deterministic software anomalies. The applied method monitors the

system and reduces the recorded features by around 60% with Lasso Regularization using Random Forest for classification. The achieved validation errors are less than 1%. In addition, alternative methods based on Decision Trees, LDA/QDA, Naive Bayes, Supported Vector Machines and K-nearest neighbors were compared as well.

As cloud infrastructure and NFV services are distributed and further producing a large amount of logs, such logs can be used for root-cause analysis of error situations in the running services[5]. The tool LOGAN inspects the divergence of current logs from a reference model and highlights logs likely to contain hints to the root cause. The paper presents the designed reference model for problem diagnosis which serves as a basis for detecting crucial log messages for the running system. Furthermore, LOGAN is able to analyze a large volume of logs and helps operators to save time during problem diagnosis by automatically highlighting high value logs.

3. Architecture for real-time anomaly detection

The suggested architecture is an extension of a typical cloud infrastructure utilized by telecommunication providers for NFV deployments. The three pillars – cloud management, cloud resources and anomaly detection pipeline are described in the following.

A continuous operation of telecommunication services can only be guaranteed if the underlying cloud platform itself is highly available. We assume OpenStack as an underlying infrastructure, as it currently viewed as the de-facto standard for open-source cloud software and is deployed in many data centers, so we were able to monitor and process real-word systems and information. OpenStack consists of a multitude of components that work together in order to deliver the desired Cloud management services. A full description of all services and their interfaces would go beyond the scope of this paper and the general architecture of OpenStack is well known[1]. To achieve high availability, every OpenStack controller service must run in a replicated fashion. At the top of the stack we place a pair of load balancers. A virtual IP address serves as an entry point for all services of the stack. The load balancers are running in an active-passive replication mode and upon failure of the active node the passive node will automatically take its place.

The replicated OpenStack installation consists of multiple instances of the controller node and some additional instances of the network node each running on separate physical machines. For load balancing, the OpenStack

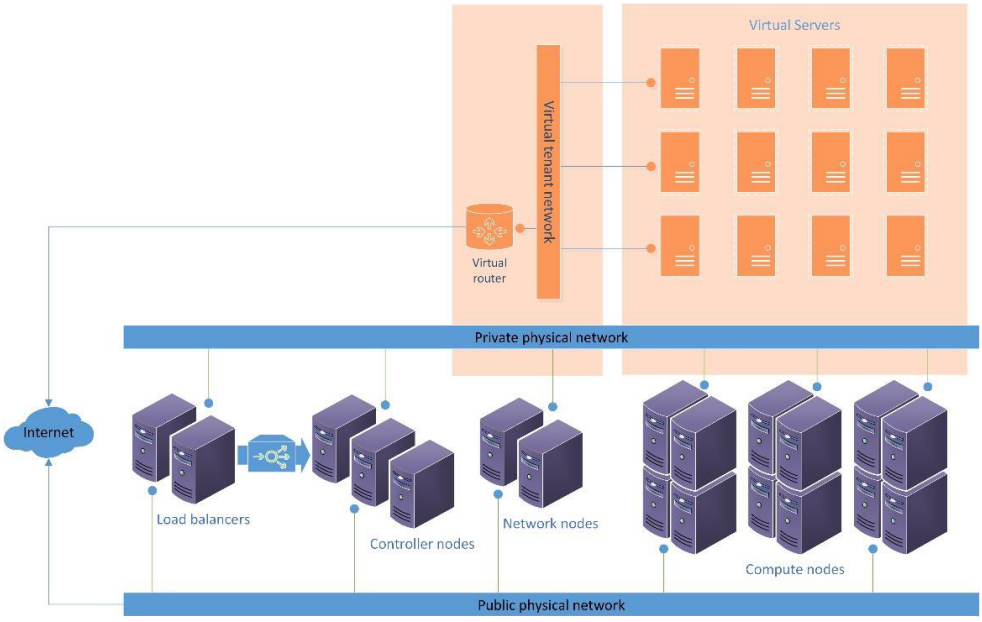

Figure 1: OpenStack service distribution over multiple physical machines

controller services rely on the native clustering capability of the deployed message queue server and the stateless property of the OpenStack controller services. The relational database used by OpenStack must be replicated as well in order to avoid data loss. The Glance image service requires shared storage, which can be provided by a distributed file system like Ceph[†]. Figur presents the OpenStack services distributed on multiple physical machines and an example NFV infrastructure deployed on top of OpenStack.

We extended the replicated OpenStack installation through additional components for monitoring the vital service parameters, for anomaly detection, and with a self-stabilization framework that contains a selection of countermeasures to mask and/or repair failures. The current setup including the cloud management component, the execution of VNF services, and the self-stabilization pipeline is depicted in Figure . In the following, the individual components of the self-stabilization pipeline are described in more detail.

The first step in this process is to collect monitoring data in all hosts and other components in the system, so the framework can identify, isolate, and evaluate indicators for the detection of anomalies. The basic data to collect represents the usage of different resources in each host. This includes CPU and RAM usage, I/O operations of different partitions and mount points, and network I/O metrics of different network interfaces and protocols. Such resource usage metrics are collected both on the physical nodes and inside of every virtual machine. In addition, the virtualization stack is queried for information. For example the virtualization library Libvirt offers an APIs to retrieve resource usage information of VMs. Open vSwitch[6] is a widespread software switch used for implementing virtual networks and offers remote APIs to retrieve network I/O metrics about its virtual bridges and ports. This list

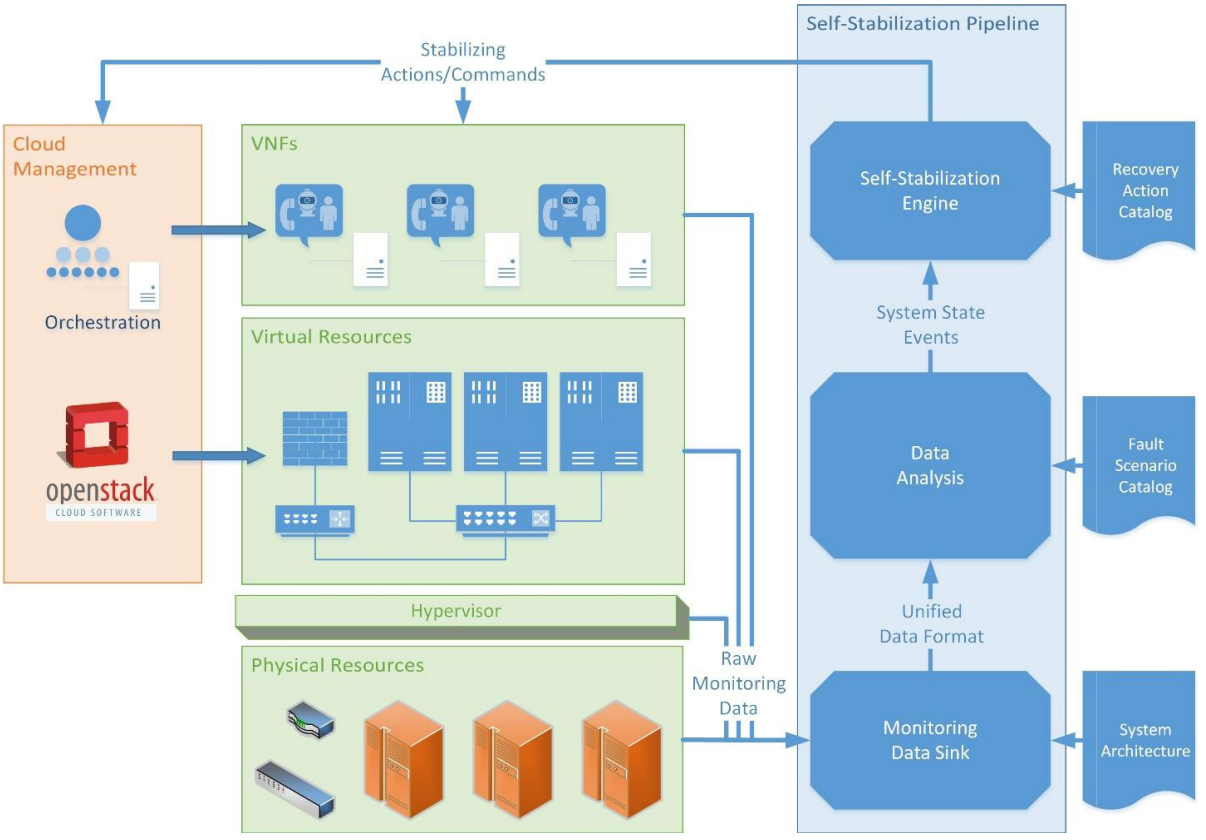

Figure 2: System architecture for anomaly detection in an NFV system

[†] http://ceph.com/

has to be extended to all subsystems and components involved in the NFV stack. An important non-functional requirement for the data collection, as well as for the entire anomaly detection pipeline, is to consume only a small portion of the systems resources. In particular, the operation of the virtualized services should not be impacted by the anomaly detection.

The monitoring data is collected in a data sink which is responsible for converting the data from all data sources into a common internal format. Information about the system architecture is used to associate incoming data streams with the correct system components.

The data analysis step receives the uniform data and condenses it into higher-level information. A variety of data analysis or data mining techniques are used here. The simplest approach is to use a complex event processor and manually define thresholds for the metrics that are known to indicate anomalies. However, this requires detailed expert knowledge to define such thresholds and even experienced administrators can miss certain patterns or dependencies hidden in the data. The major challenge for the failure prediction and failure recovery is related to the definition of sets of characteristic values that indicate the presence of an anomaly or failure. Such feature vectors can be engineered top-down based on anticipated behavior of the system in case of failures. However, such a systematic approach is less visible for large, productive environments, so usually a bottom-up approach – inserting failures and measuring the system parameters – is applied. The collection of such feature vectors serves as a starting point for a similarity search in order to detect and then recover failures in a running server. Thus, the main challenge of the data analysis step is to develop and deploy corresponding retrieval mechanisms, which can separate the noise in a running infrastructure from the current processing state and detect anomalies as precisely as possible.

Therefore, we developed and implemented advanced techniques based on online unsupervised clustering and classification algorithms capable of handling continuous data streams. Multiple analysis steps are be chained together and executed on different hosts to achieve scalability. For example, every virtual and physical host performs an analysis of all locally collected data and forwards these intermediate results to a higher-level analysis which combines all information for a group of hosts and finally forwards its own results to a global analysis step which performs root cause analysis using the distilled information about the entire NFV system. All data analysis steps have access to a catalog of known anomalies to improve the analysis results.

The output of the data analysis are system state events sent to the self-stabilization engine. The purpose of this engine is to execute recovery actions in the cloud management platform or directly within the NFV resources. Pre-defined actions from the recovery action catalog are used to guide the system back to a “healthy” state. The recovery action catalog can include recipes for an orchestration tool to run actions on the service layer or the underlying cloud system. Possible actions include migrating a VM to another hypervisor or changing the configuration of an OpenStack service such as a DHCP agent in Neutron. Recovery actions can be combined and executed on different layers simultaneously. Through the continuous data monitoring and analysis, the decision engine will receive continuous feedback on the success of the implemented recovery routines. Based on that it can execute additional recovery or mitigation routines or contact an administrator as a last resort.

4. Prototype

Several parts of the architecture are already implemented on a dedicated testbed of 20 physical machines. The basis is an installation of OpenStack Liberty with a three-fold replicated controller node and a two-fold replicated network node. A pair of replicated load balancers provides a reliable access to all OpenStack services. The remaining nodes are used as compute nodes hosting the virtual machines. To simulate the workload of an NFV installation, the open source IMS core implementation Project Clearwater[‡] is executed on top of OpenStack. OpenStacks own orchestration module Heat is used to deploy the virtual infrastructure, and the lightweight configuration management tool Ansible[§] configures all virtual machines to run their respective VNFs.

[‡] http://www.projectclearwater.org/

[§] https://www.ansible.com/

The data collection is implemented in the Go programming language and collects between 130 and 180 metrics on a typical Linux machine. The data is mainly obtained by parsing the `/proc` file system in short time intervals. On hypervisor nodes, additional 22 metrics are collected for every hosted virtual machine by querying the Libvirt API and 7 metrics for every virtual network interface inside Open vSwitch. This adds up to about 500 metrics on a small sized compute node. When collecting the data in 300ms intervals and sending it over the network in a dense binary format the CPU usage does not exceed 3% of a single 3.3 GHz processor. Since many of the metrics do not change frequently, the sampling rate could be selectively lowered to further reduce the resource overhead.

Preliminary evaluation of the collected data indicates a high degree of reliable recognition of pre-defined failure scenarios, exceeding 95%. These results need to be further investigated in terms of the execution environment and the impact of production "noise" – concurrently running processes, user interaction, and varying load – in order to make a reliable statement on the efficiency and the precision of the framework. This is part of the future work described in the following.

4. Conclusion and future work

This paper presents a set of methods, an implemented prototype, and preliminary evaluation for anomaly detection in cloud-based infrastructure with specific focus on deployment of virtualized network functions (VNF). The framework is built upon OpenStack, which is the current de-facto standard in open-source data center cloud software. Our system architecture targets the NFV use case as the demands for fault tolerance are especially high in the case of telecommunication providers, which are used to execute services on dedicated and specialized hardware. However, the flexibility and the cost effectiveness of virtualized solutions motivate service providers to deploy virtualized solutions and simultaneously research software-based methods for increasing their fault tolerance. The presented research aims at solving this problem by providing an extensive monitoring solution that collects a significant set of data and analyzes it with supervised and non-supervised machine learning techniques. The computed indicators are used to activate pre-defined countermeasures to mask or even repair outages or situations with degraded performance.

In the next steps the testing environment will be extended by components for injecting general fault scenarios like overload or memory leaks, together with fault scenarios specific to NVF environments. Based on the collected data, additional unsupervised machine learning algorithms will be implemented inside the data analysis framework. Further, a visualization of the current system state will help to evaluate the success of the anomaly detection and to find correlations between different system layers. New cloud-related optimization technologies introduce layer violations, which further exacerbate anomaly detection in NFV infrastructures. Such technologies, like the DPDK[**] will be taken into consideration and further set our solution apart from conventional monitoring and fault detection systems.

References

1. OpenStack [Online]. Available: https://www.openstack.org/.
2. Prathamesh Gaikwad: "Anomaly Detection for Scientific Workflow Applications on Networked Clouds" in *2016 International Conference on High Performance Computing & Simulation* (*HPCS)*, Insbruck, Austria.
3. Alessandro Pellegrini, Pierangelo Di Sanzo, Dimiter R. Avresky: "A Machine Learning-Based Framework for Building Application Failure Prediction Models," in *2015 IEEE International Parallel and Distributed Processing Symposium Workshop (IPDPSW),* Hyderabad, India.
4. Javier Alonso, Lluis Belanche, Dimiter R. Avresky: "Predicting Software Anomalies Using Machine Learning Techniques," in *2011 10th IEEE International Symposium on Network Computing and Applications (NCA),* pp. 163-170.
5. B. C. Tak, S. Tao, L. Yang, C. Zhu and Y. Ruan: "LOGAN: Problem Diagnosis in the Cloud Using Log-based Reference Models," in *2016 International Conference on Cloud Engineering* (*IC2E),* Berlin, Germany.
6. Ben Pfaff, Justin Pettit, Teemu Koponen, Ethan Jackson, Andy Zhou, Jarno Rajahalme, Jesse Gross, Alex Wang, Joe Stringer, and Pravin Shelar: "The Design and Implementation of Open vSwitch," in *12th USENIX Symposium on Networked Systems Design and Implementation*, Oakland, CA, USENIX Association, 2015, pp. 117--130

[**] http://www.dpdk.org/