Demo: A Deep learning based SLA management for NFV-based services

Jaafar Bendriss,
Imen Grida Ben Yahia
Orange Labs
Chatillion, France

Roberto Riggio
CreateNet
Pisa, Italy

Djamal Zeghlache
Telecom SudParis
Every, France

ABSTRACT

Network function Virtualization (NFV) is expected to accelerate service deployment and enable service agility for Telco operators. NFV builds on virtualization technology and off-the-shelf general purpose hardware, shifting previously dedicated Telco appliances towards the cloud ecosystem. This paradigm shift brings new concepts such as dynamic service chaining that necessitates to rethinking the Network Management approach. In this paper we present CogSLA, a data-driven solution for SLA enforcement in an NFV deployment. CogSLA uses Deep Feedforward Neural Network or Multi-Layer perceptron (MLP) for achieving proactive identification of SLA violations. The key contribution of this work is to proactively change the network state, anticipating and avoiding foreseeable SLA violation.

Categories and Subject Descriptors

H.5.m. [**Network Management**]: NFV; H.5.m. [**AI**]: Artificial Neural Networks

Keywords

NFV,ANN, Network Function Virtualization

1. INTRODUCTION

Telecommunication operators are highly concerned by the carrier-grade specifications, i.e. high availability 99.999% and high performance delivery. Migrating the network functions to low-cost, non-highly reliable servers while maintaining carrier-grade service quality is considered as the biggest challenge facing the NFV adoption. Two approaches emerged for tackling this challenge, (1) *overprovisioning* the network to reduce the risk of service degradation while bounding to the predefined SLA, (2) *cognitive management* - as shown in Figure1 - that monitors in near real-time the network and anticipates service degradation based on the SLAs. We adopt the latter approach.

The limitation of current monitoring tools and approaches is that they are passive; they rely on thresholds, waits for alerts and notification to identify service degradation. However, this approach is not fully scalable because one cannot induce easily from a flood of notifications and alerts the source of the errors or the necessary corrective actions. Moreover, VNFs can enter in abnormal states even though all the underlying resources are behaving normally which constitute a hard problem that require more new data sources (at the application/service level) and/or effective extrapolation from system-level metrics. Likewise, NFV management should consider the service logic defining the ordering of VNFs interconnections that delivers network services.

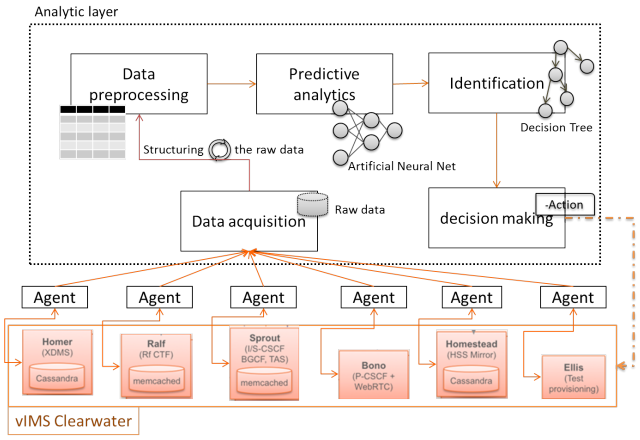

Figure 1: CogSLA NFV Management Framework

In this demo, we present a deep learning-based SLA management platform for NFV-based services. Therefore, we will show, (Figure 1) the data acquisition (data collection and monitoring), preprocessing (data cleansing and transformation), analysis (forecasting and identification) and execution (decision making). More specifically, we will demonstrate the identification of SLA violation of VOIP service on top vIMS (virtual IP Multimedia Subsystem) platform. The violation forecasting is made with an ANN and the identification with Decision Tree techniques, all in proactive manner.

2. THE DEMO

Our Demo can be divided into two parts (see Figure 1), the test bed and the analytic layer. The test bed is The Clearwater VNF set up using Openstack as an infrastructure manager, this is shown as a webpage with all the instances.

DOI: 10.1145/1235

The VNF configuration is based on six Clearwater's VM (VNF Component), a DNS bind VM for load balancing and namespace management, an central monitoring VM and a stress VM responsible for the service degradation. To demonstrate how the traditional approach is made by showing the Openstack monitoring service, Monasca (Monitoring-at-scale), how alerts and notifications are configured, and the visualization dashboard, Graphana. Next, we show on Tableau Software a guided step-by-step process of data analysis. These steps are developed in the subsequent paragraphs from a data-centric perspective[1].

2.1 Data generation: Monitoring the test bed

The use case studied for NFV management is an open source vIMS (virtual IP Multimedia Subsystem) Clearwater that uses SIP as a call control for voice and video communications [cle]. Clearwater is compliant with the basic IMS architectural principles and interfaces well-known in the telecommunication world while implementing a cloud-oriented design that is used by elastic cloud architectures. Clearwater is a VNF based on 6 Virtual Machines (VM) each of which implements a component of the vIMS, also termed by the ETSI as VNF Component (VNFC). The VNF is monitored by installing in each VM a monitoring agent that collects the data locally and push it back to a central monitoring server. The monitoring frequency is set at 30 seconds and the for each VM, we collect 30 system-level variables. The Service degradation problem is generated by SIPp tools, generating 30.000 new clients to stress testing the VNF.

2.2 Data acquisition: Data collection through REST API

We used Monasca REST API to access the data in batches. The input is a JSON string containing all the database data. We process these data by filtering irrelevant information. Then, we transformed the relevant data into multiple tables containing key-value pairs corresponding to timestamp-metric. Next, we regroup all the tables into a general matrix with key-values tuples as timestamp interval to metrics. From now on we consider the metrics as naive (i.e. unprocessed) features. In the demo, we show how the inputted JSON string is transformed to an exploitable table.

2.3 Data preprocessing: From naive to feature engineering

From the outputted table in the data acquisition module, we construct a second table based on combination of features as shown in Table 1. The data preprocessing phase aims at improving the data quality in terms of structure, noise and consistency. Moreover, this phase acts as a requirement checklist between the data sources and the analytical framework.

Data cleansing and transformation. We remove corrupted values, redundant entries and null or undefined metrics. This dramatically improves the data quality and is a necessary step that takes multiple iterations and analysis before complete automation. Afterwards, the time series are transformed into a stationary state with a mean of 0 and a standard variation of 1, to avoid gradient explosion while training the ANN. The data cleansing and transformation is performed by a Python script.

[1]The source code and the video are available in GitHub at https://github.com/heekof/CogSLA.

Features	Rule
Feature 1	`cpu.user_perc`^{2} / `mem.usable_perc`
Feature 2	`mem.usable_perc`
Feature 3	`net.out_packets_sec` / `mem.usable_perc`
Feature 4	`net.out_packets_sec` / `disk.sp_used_perc`

Table 1: Feature Engineering Rules

Data visualization. The visualization is a method that allows drawing insight from the data. For more in-depth analysis techniques, we programmed in Python Seaborn library, we used autocorrelation as a pre-test on the data for determining whether they are forecastable or not. If yes for how many steps. Pearson correlations of all monitored variables to allow us to track most correlated pairs, thus reducing the dimensionality of the variables and detecting anomalies when the correlations are broken. Finally, we demonstrate the use of PCA to reduce effectively the dimensionality. We will present in the demo using Tableau Software and Jupyter notebook.

2.4 Data analytics: Forecasting and identification

We trained our Feedforward Neural Network with 1500 epochs. However, in this demo, we will show the training process for a limited sample size with 500 epochs. An epoch corresponds to the FFNN runs through all the training data set. We used Google's TensorFlow library for the ANN design and training.

Training. In offline mode, we prepare the data for the supervised learning. Prior to fitting the model we create two series lagged by 1-step. The observation of the previous time step corresponds to the current time step. The ANN training uses backpropagation algorithm to optimize the neural weights finding the coefficients that captures the best relationship between the past and the future. The learned model is stored as a 2D-matrix for the online phase.

Forecasting. Loading the learned ANN model and executing on the four features in Table 1, will result in forecasted values for these inputs. For sake of simplicity, we will show in the demo in a graph how the monitored `net.in_packets_sec` evolves and its predicted values in another graph

Detection. We used Decision Tree algorithm IR3, that takes as inputs the 4 ANN outputs (forecasts) and to classify them it into SLA classes (i.e. SLA violation, non-SLA violation)

2.5 Decision making: Execution

In this example, we set the network management action as a corrective Openstack action that shuts down the VM responsible for the overload. The corrective action uses Openstack Nova API. Nova API service allows the programmability of the Cloud compute service. As an example, we demonstrate during the demo how our system reacts to drop on the service quality by predicting the evolution of network input metric `net.in_packets_sec`.

3. REFERENCES

[cle] Welcome to Clearwater - Project Clearwater 1.0 documentation. (????). https://clearwater.readthedocs.io/en/stable/

Demo requirements.

Equipment to be used for the demo : 2 screens
Space needed: 4 m2
Setup time required: 10min
Additional facilities: Wifi access, water