Estimating VNF Resource Requirements Using Machine Learning Techniques

Houda Jmila(✉), Mohamed Ibn Khedher, and Mounim A. El Yacoubi

SAMOVAR, Telecom SudParis, CNRS, University of Paris-Saclay,
9 rue Charles Fourier, 91011 Evry Cedex, France
{houda.jmila,mohamed.ibn_khedher,mounim.el_yacoubi}@telecom-sudparis.eu

Abstract. Resource Management in the network function virtualization (NFV) environment is a challenging task. The continuously varying demands of virtual network functions (VNF) call for dynamic algorithms to efficiently scale the allocated resources and meet fluctuating needs. In this context, studying the behavior of a VNF as a function of its environment helps to model its resource requirements and thus allocate them dynamically. This paper investigates the use of machine learning techniques to estimate VNFs needs in term of CPU as a function of the traffic they will process. We propose and adapt a Support Vector Regression (SVR) based approach to resolve the problem. Results show its efficiency and superiority compared to the state of the art.

Keywords: Virtual network function · Resource management · Machine Learning · Support Vector Regression

1 Introduction

Applying Machine Learning (ML) techniques to control and operate networks is a promising and attractive research field. Recent advances in network architecture and telemetry [1] facilitate the modeling and implementation of such solutions. Specifically, the Software Defined Network (SDN) paradigm [2] decouples the "data plane" from the "control plane" and thus enables central control of the network. Moreover, current data plane elements (routers, switches etc.) are equipped with more powerful storage and computing techniques and hence can provide a richer global view of the network. These conditions ease and ameliorate learning about the network to better supervise it. Recently, Mestres et al. [3] introduced a "Knowledge Defined Networking architecture" explaining how such an ecosystem should operate to provide automated network control. The authors briefly investigated some use cases where basic ML techniques are applied to solve networking problems.

One of the interesting investigated issues is that of allocating resources [4] in the Network Function Virtualization [5] context. NFV is a recent paradigm that decouples the network functions (such as firewalls, proxies, Network Address Translators (NATs) etc.) from the physical equipments on which they run.

These functions, decoupled from the underlying hardware and known as Virtualized Network Functions (VNFs), run on virtualized resources (e.g. Virtual Machines (VMs)) and are usually connected together to form a service function chain (SFC) that supports a required service (for example a video Content Delivery Network (CDN) transporting live and on-demand videos to end users).

Resource management in NFV is a challenging problem considering the scarcity of the resources allocated to VNFs [6]. The optimal placement and chaining of VNFs has been extensively investigated [7] when the network is static. However, the dynamic real environment calls for algorithms able to continuously scale the amount of resources allocated to VNFs to process the fluctuating traffic passing through them (for example filter the traffic or detect intrusion). Studying the behavior of a VNF as a function of its environment (virtual network topology, traffic characteristics, current configuration, etc.) helps modeling its resource requirements (in terms of CPU, memory, storage etc.) and thus allocating them automatically and efficiently (do not allocate more or less than needed). However, the behavior of VNFs is dynamic, complex and depends on different factors, which makes developing accurate models a challenging task. In this context, machine learning is a promising way to achieve this goal. This paper investigates the use of machine learning solutions to estimate virtual network functions CPU requirement as a function of the traffic they process. This problem was briefly introduced in [3] but requires more in-depth study. The contribution of this paper is twofold. First we give a comprehensive survey of existing machine-learning approaches for resource management in NFV. Second, a support vector regression based model is designed and adapted to optimally determine CPU consumption for different virtual network functions. To the best of our knowledge, this is the first work highlighting such contributions.

The paper is structured as follows. Section 2 presents an expanded survey on machine learning applications for resource management in NFV. Section 3 defines the tackled problem. The proposed approach is described in Sect. 4 and evaluated in Sect. 5. Finally, Sect. 6 concludes the paper.

2 State of the Art

Machine learning is a type of artificial intelligence that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. The use of machine learning techniques in NFV is a research field that requires more attention. This section regroups the current approaches considered in the domain. Its purpose is to inspire researchers to new applications of machine learning and ameliorate actual models. We will briefly introduce the NFV resource allocation issue, then present machine learning based solutions to resolve variants of the problem.

In NFV, services are composed of one or more VNFs connected in a specific order to create a Service Function Chain (SFC) supporting the service. Each VNF requires an amount of resources to process the traffic passing through it. The top of Fig. 1 shows an example of SFC composed of three VNFs

(Firewall, Intrusion Detection System, Proxy). To deploy a SFC, an operator needs to find the right placement of VNFs into the nodes (virtual machine, container, etc.) of the physical network having enough available resources. Various constraints like the scarcity of physical resources and the need to respect the service Level of Agreement (SLA) should be taken into account. Different metrics as the mapping cost and the algorithm rapidity can be considered during the placement process. An example of such a mapping is displayed in Fig. 1.

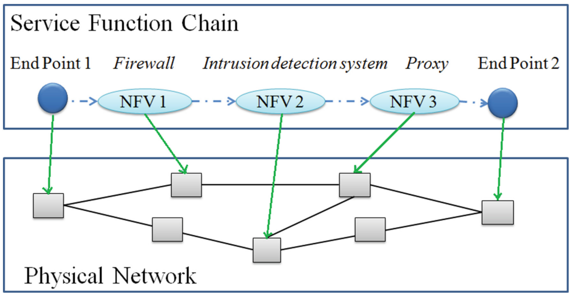

Fig. 1. An example of service function chain mapping

Once the hosts are selected, the required virtual resources are created and booted to instantiate the VNFs. The VNFs are then chained through a physical path to steer the traffic flowing between them. This traffic can fluctuate dynamically during the service lifetime. To continue processing it, the VNFs resource requirements may vary (increase or decrease) over time. A good resource allocation algorithm should continuously scale the allocated resources to meet new VNF demands. Such elasticity can be performed by allocating/releasing resources to/from the VNFs (vertical scaling) or by removing/migrating the VNFs (horizontal scaling). Allocated resources are finally freed when the lifetime of VNFs expires.

Mijumbi et al. [8,9] addressed the problem of dynamically allocating and managing resource fluctuation on already mapped VNFs during their lifetime. They designed a Graphic Neural Network (GNN) [10] based model to predict the VNFs requirements. They argued that there is a non-negligible delay in spinning-up (create, boot and instantiate) new resources, so determining resource needs ahead of time would avoid system outages and QoS degradation. The authors mentioned that resource requirements of a VNF depend on those of its neighboring VNF since traffic flows between them. This dependency motivated them to use the connectionist approach derived from the GNN to predict each VNF requirements by observing its historical resource utilization and those of its neighbors. To do so, each VNF n was described by a feature vector f_n representing its actual and past required resources (memory m_n, CPU c_n, and processing delay d_n) included in a finite time horizon π:

$$f_n(t) = \left[c_n(t)\ m_n(t)\ d_n(t)\ \dots\ c_n(t-\pi)\ m_n(t-\pi)\ d_n(t-\pi)\right]^T \tag{1}$$

A GNN model was then designed to represent the star topology formed by the VNF and its attached VNF neighbors. Feed forward Neural Network functions (FNN) were applied in different GNN layers to compute the VNF demand. To evaluate their proposal, the authors used an open source IMS (IP Multimedia Subsystem) core named "Clearwater" [11] which provides SIP (Session Initiation Protocol)-based call control for voice and video communications. After the training period, the prediction accuracy was evaluated and results showed that GNN performs better than a classic FNN. Moreover, dynamically allocating resources based on requirement prediction improved the calls acceptance rate.

The authors of [12] examined the VNF placement and chaining while minimizing the end-to-end latency. They explained that the solution of VNF placement based on calculations at time t can be inappropriate for time $t+1$ where the VNF are effectively placed. This is due to the resources spinning-up process that can take a long time; thus the physical network state (available resources, geographic location, etc.) may change meanwhile. The authors proposed to predict the state at $t + 1$ so that the placement remains consistent with the requirements. To do so, they used a Support Vector Regression model to forecast the delay between two end points at time $t+1$. They proposed a training vector that captures the parameters affecting the delay: (*i*) the inference caused by resource sharing on end nodes, (*ii*) the length of the link connecting them and (*iii*) the traffic passing through it. The authors developed a C++ simulator to generate a training dataset that involves different node and link queuing delays. To achieve this, they used a Stochastic modeling for delay analysis of a VoIP network [13]. The results showed that the system predicts the delay rapidly and increases the number of successful service chains embedding.

In [14], the authors applied a Bayesian learning method to predict the reliability of cloud resources based on their historical usage. Reliability represents the ability of a resource to ensure constant system operation without disruption. The prediction result was used to improve the performance of a Markov Decision Process used to allocate VNFs on demand in a cost effective manner.

In [15], the authors used machine learning to identify the most appropriate types of resources to be allocated to a VNF regarding workload characterization. They constructed their own database by measuring the VNF performance under different deployment configurations. Using the C4.5 algorithm [16], a decision tree was then generated to relate performance indicators to the deployment configuration and to select the best one.

The authors of [3] introduced the idea of knowledge defined networking and an architecture enabling the management of networks using machine learning knowledge. They briefly examined some use cases including the VNF resource allocation problem. Mestres et al. used an Artificial Neural Network (ANN) based solution to predict the CPU resource demand given the entering traffic. Their approach will be compared to ours in Sect. 5.

In conclusion, we notice that the ML techniques were used to predict one or more VNF placement constraints (required resources, latency (QoS), resource reliability etc.) in order to enhance and accelerate the resource allocation algorithm performance. In this paper, as in [8], we focus on predicting VNF resource needs to dynamically scale the allocated resources. Unlike Mijumbi et al. [8] who exploited historical usage for prediction, we estimate VNF resource consumption depending on the traffic to be processed. The next sections detail the problem.

3 Problem Definition and Proposed Approach

The considered question is the following: given an incoming traffic entering the VNF, the objective is to determine the amount of CPU required by the VNF to process that traffic. The CPU's need may depend on many known and unknown interacting parameters. This has motivated us to use a ML based technique. Once the prediction is performed, the CPU's estimated value can be used by a resource allocation algorithm [7] to automatically adapt the amount of provisioned resources, as shown in the left of Fig. 2.

To solve the described problem, we propose an approach based on a supervised ML technique called Support Vector Regression (SVR) [17]. As shown in the right of Fig. 2, the main idea is to use a training dataset to learn/train an SVR estimation function/model able to predict a CPU value for each entering traffic. Such a dataset should be constructed ahead based on network telemetry and measurement and should contain pairs of (traffic vector, CPU), where traffic vector describes the incoming traffic and CPU is the amount of resources consumed to process it (i.e. the ground truth). To represent the traffic, various features like the number of packets, the total bytes, the source and destination ports and IP, etc., are extracted off-line in 20 s batches. To reduce the features dimensionality and thus accelerate the prediction, we apply a Principal Component Analyses (PCA) [18] method. The CPU consumption level is then predicted

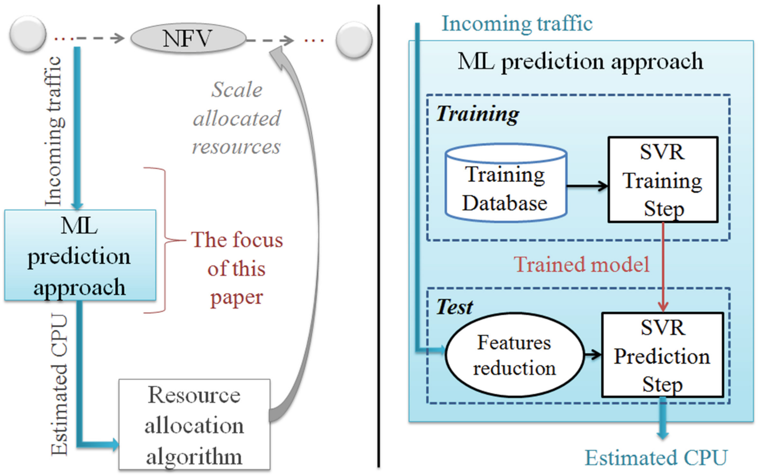

Fig. 2. Flowchart of our approach

using the pre-trained estimation function. The details about the designed solution are given in the next section.

4 CPU Consumption Estimation Based on SVR

SVR is a supervised learning model used for data regression analysis. SVR is able to separate data in a large dimension space, non-linearly, by applying kernel functions [19]. This capacity motivated us to apply a SVR predictor to model the noisy traffic data in the NFV environment. To accelerate the prediction, we use Principal Component Analysis for dimension reduction. In the following paragraphs, we describe respectively (*i*) the use of PCA, (*ii*) the principles of an SVR predictor and (*iii*) its adaptation to our problem.

4.1 Features Dimensionality Reduction

PCA is a multivariate statistical technique used in many scientific disciplines. Its goal is to extract the important information from the data and to represent it as a set of new orthogonal variables called principal components. To do so, a linear mapping of the data to a lower-dimensional space is performed to maximize the variance of the data in the low-dimensional representation. This can decrease the algorithm execution time without necessarily degrading its performance. In this paper, we retain 95% of data variance to conserve a maximum of traffic information.

4.2 Principles of Prediction Based on SVR

Given a training data S, composed of N input d-dimension vectors $\mathbf{x_i}$ with corresponding target values y_i. $S = \{(\mathbf{x_i}, y_i)_{i,1\leq i\leq N}, \mid \mathbf{x_i} \in R^d, y_i \in R\}$. The idea of the regression problem is to estimate an objective function $f : \mathbf{x_i} \longmapsto y_i$ able to predict an output value from a query input. The generic form of the support vector regression function can be stated as:

$$f(x) = \mathbf{w} \cdot \phi(\mathbf{x}) + b \tag{2}$$

where $\mathbf{w} \in R^n$, $b \in R$ and ϕ denotes a mapping function from R^n to a high dimensional space. At this step, the goal is to find the value of $\mathbf{w}$ and b by minimizing the empirical regression risk R given in the following equation:

$$R(f) = \frac{1}{N}\sum_{i=1}^{N} \Gamma((f(x_i), y_i)) \tag{3}$$

where, Γ is an ϵ - insensitive loss function that measures the empirical risk.

4.3 SVR for CPU Demand Prediction

For a given VNF, a SVR predictor is learned (resp. evaluated) on the training dataset (resp. test dataset). This involves two steps: (*i*) model construction and (*ii*) CPU prediction.

Model Construction. The SVR model is estimated using a training dataset composed of N samples $S_i, 1 \leq i \leq N$ of couples $(\mathbf{TRF_i}, CPU_i)$. $\mathbf{TRF_i}$ is a vector describing the entering traffic, with features of dimension d measured frequently. CPU_i is the amount of CPU used by the VNF to process the traffic $\mathbf{TRF_i}$. Using $S_i, 1 \leq i \leq N$, the SVR model is estimated by computing the objective function in Eq. 2.

CPU Prediction. For a newly arriving traffic vector, PCA is first applied to reduce its dimension. Then, the corresponding CPU value is predicted using the pre-learned objective function as following:

$$CPU = f(\mathbf{TRF}) = \mathbf{w} \cdot \phi(\mathbf{TRF}) + b \tag{4}$$

5 Experimental Results

5.1 Database Description

To asses the performance of our solution, we made extensive evaluation tests on a dataset provided in [3], where CPU consumption of real-world virtual network functions are measured in 20 s batches when operating under real traffic. Each entering traffic is described by 86 features. Three VNFs were considered: (*i*) an SDN-enabled switch (*ii*) an SDN-enabled firewall and (*iii*) a snort [20]. Both VNFs (*i*) and (*ii*) were implemented using Open VSwitch [21]. The first VNF, a switch, receives, processes, and forwards data to destination devices. The second, a firewall, controls the incoming and outgoing network traffic based on predetermined security rules and finally, the snort function is an Intrusion Detection System. Table 1 summaries the number of available samples per VNF.

Table 1. Database description

Virtual network function	Number of samples (N)
Firewall	755
Switch	1172
Snort	1359

5.2 Evaluation Protocol

For the evaluation, we operated 100 runs for each VNF. In each run, 80% of the available data is selected randomly and used for training and the resting 20% for test. For an iteration i, the Prediction Error (PE_i) is calculated as follows:

$$PE_i = \frac{1}{N} \sum_{j=1}^{N} (cpu_j - cpu_j{}^*) \tag{5}$$

where N is the number of samples, cpu_j is the real (ground truth) CPU of the j^{th} test sample, and the $cpu_j{}^*$ is the predicted value. The Average Prediction Error (APE) over 100 iterations is:

$$APE = \frac{1}{100}\sum_{i=1}^{100} PE_i \tag{6}$$

The average prediction error (APE) is used as the main metric to compare the different prediction techniques presented below.

5.3 Results

We compared our results with those of [3] where a one hidden layer Artificial Neural Network (ANN) is used for CPU prediction. We present the results in different scenarios where we apply (or not) PCA to evaluate their contribution. Tables 2 and 3 show the obtained results.

Table 2. Results of CPU predictions using machine learning

ML methods	Average prediction error (APE)		
	Firewall	Switch	Snort
ANN	3.71 ± 0.012	5.26 ± 0.015	15.11 ± 0.019
ANN + PCA	4.04 ± 0.1	5.80 ± 0.01	17.63 ± 0.019
SVR	**2.65 ± 0.003**	**3.76 ± 0.004**	**13.32 ± 0.015**
SVR + PCA	3.24 ± 0.004	4.69 ± 0.006	15.46 ± 0.017

The first remark is that all machine learning techniques accomplished a reasonable prediction performance with an error less then 18 %. This demonstrates the efficiency of ML in solving this estimation problem. Second, independently of the ML used technique, we note that the APE changes significantly from one VNF to another (except for firewall and switch which are implemented with the same technique and thus have the same behavior). This shows that the ML performance depends extremely on the VNF nature and its correlation with its environment. Specifically, using only the entering traffic as a factor to estimate the CPU is not always reliable. This is shown by the APE for snort that did not fall under 13%. This may be due to other factors in the VNF environment that are influencing the CPU consumption and should be analyzed to accomplish better prediction.

Third, the results show the superiority of SVR over ANN. Moreover SVR is more stable with only 0.003 of standard error. This is expected as the ANN has only one hidden layer, and the available dataset does not allow considering a deep neural network with more hidden layers. Finally, to evaluate the contribution of PCA, we measure the average time spent to estimate a CPU given its

Table 3. Contribution of PCA

ML methods	Average execution time
SVR	208 µs
SVR + PCA	40 µs

entering traffic when applying, or not, PCA (Table 3). When looking jointly to Tables 2 and 3, we see that PCA reduces five times the execution delay without significantly degrading the prediction performance (less than 2% of APE degradation). It is to be expected that these results would be more remarkable when the number of features describing the traffic gets much larger.

6 Conclusion

In this paper, we examined the use of machine learning techniques in the virtual network function domain and studied the example of estimating the CPU usage based on traffic characterization. The obtained results demonstrate the efficiency of such solutions. Note that this performance can be improved further using efficient deep learning methods for large scale networks and abundant data. In future work, we plan to design a dynamic resource allocation framework and test the benefit of using ML based demand prediction in achieving elastic allocation.

References

1. Kim, C., Sivaraman, A., Katta, N., Bas, A., Dixit, A., Wobker, L.J.: In-band network telemetry via programmable dataplanes. In: SIGCOMM Industrial Demo Program (2015)
2. Kreutz, D., Ramos, F.M., Verissimo, P.E., Rothenberg, C.E., Azodolmolky, S., Uhlig, S.: Software-defined networking: a comprehensive survey. Proc. IEEE **103**(1), 14–76 (2015)
3. Mestres, A., Rodriguez-Natal, A., Carner, J., Barlet-Ros, P., Alarcón, E., Solé, M., Muntés, V., Meyer, D., Barkai, S., Hibbett, M.J., et al.: Knowledge-defined networking (2016)
4. Jmila, H., Drira, K., Zeghlache, D.: A self-stabilizing framework for dynamic bandwidth allocation in virtual networks. In: 2016 IEEE/IFIP Network Operations and Management Symposium, pp. 69–77 (2016)
5. Mijumbi, R., Serrat, J., Gorricho, J.L., Bouten, N., Turck, F.D., Boutaba, R.: Network function virtualization: state-of-the-art and research challenges. IEEE Commun. Surv. Tutor. **18**(1), 236–262 (2016)
6. Mijumbi, R., Serrat, J., Gorricho, J.L., Latre, S., Charalambides, M., Lopez, D.: Management and orchestration challenges in network functions virtualization. IEEE Commun. Mag. **54**(1), 98–105 (2016)
7. Herrera, J.G., Botero, J.F.: Resource allocation in NFV: a comprehensive survey. IEEE Trans. Netw. Serv. Manag. **13**(3), 518–532 (2016)

8. Mijumbi, R., Hasija, S., Davy, S., Davy, A., Jennings, B., Boutaba, R.: A connectionist approach to dynamic resource management for virtualised network functions. In: 12th International Conference on Network and Service Management (CNSM), pp. 1–9. IEEE (2016)
9. Mijumbi, R., Hasija, S., Davy, S., Davy, A., Jennings, B., Boutaba, R.: Topology-aware prediction of virtual network function resource requirements. IEEE Tran. Netw. Serv. Manag. **14**(1), 106–120 (2017)
10. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE Trans. Neural Netw. **20**(1), 61–80 (2009)
11. Clearwater project. http://www.projectclearwater.org/
12. Gupta, L., Samaka, M., Jain, R., Erbad, A., Bhamare, D., Metz, C.: COLAP: a predictive framework for service function chain placement in a multi-cloud environment. In: IEEE 7th Annual Computing and Communication Workshop and Conference, pp. 1–9 (2017)
13. Gupta, V., Dharmaraja, S., Arunachalam, V.: Stochastic modeling for delay analysis of a VoIP network. Ann. Oper. Res. **233**(1), 171–180 (2015)
14. Shi, R., Zhang, J., Chu, W., Bao, Q., Jin, X., Gong, C., Zhu, Q., Yu, C., Rosenberg, S.: MDP and machine learning-based cost-optimization of dynamic resource allocation for network function virtualization. In: IEEE International Conference on Services Computing, pp. 65–73 (2015)
15. Riccobene, V., McGrath, M.J., Kourtis, M.A., Xilouris, G., Koumaras, H.: Automated generation of VNF deployment rules using infrastructure affinity characterization. In: 2016 IEEE NetSoft Conference and Workshops (NetSoft), pp. 226–233 (2016)
16. Salzberg, S.L.: C4.5: programs for machine learning by J. Ross Quinlan. Mach. Learn. **16**(3), 235–240 (1994). Morgan Kaufmann Publishers Inc. 1993
17. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**, 273–297 (1995)
18. Jolliffe, I.: Principal Component Analysis. Springer, New York (1986). doi:10.1007/b98835
19. Khedher, M.I., El Yacoubi, M.A.: Two-stage filtering scheme for sparse representation based interest point matching for Person re-identification. In: Battiato, S., Blanc-Talon, J., Gallo, G., Philips, W., Popescu, D., Scheunders, P. (eds.) ACIVS 2015. LNCS, vol. 9386, pp. 345–356. Springer, Cham (2015). doi:10.1007/978-3-319-25903-1_30
20. Snort. https://www.snort.org/
21. OVS. http://openvswitch.org/