Understand Your Chains and Keep Your Deadlines: Introducing Time-constrained Profiling for NFV

Manuel Peuster
Paderborn University
manuel.peuster@uni-paderborn.de

Holger Karl
Paderborn University
holger.karl@uni-paderborn.de

***Abstract*—Fully-automated resource dimensioning is one of the key requirements for agile, DevOps-enabled network function virtualization (NFV) scenarios in which new service versions are continuously delivered and deployed to production. To enable and support these dimensioning processes, a priori knowledge about the performance behavior of the deployed service function chains (SFC) is collected using profiling solutions that automatically generate so-called SFC performance profiles. A challenge in those profiling processes is the huge configuration space of typical SFCs that need to be explored to collect enough information to accurately describe the performance behavior of the profiled SFC.**

In this paper, we introduce the concept of *time-constrained profiling (T-CP)* which profiles only a small subset of all possible SFC configurations and uses machine learning techniques to predict performance values for the remaining configurations to generate a full SFC performance profile within a given time limit. Using our novel, open-source T-CP prototype, we analyze the accuracy of the generated profiles using different selection algorithms to find good configuration subsets. We base parts of this analysis on real-world SFC performance measurements, which we make publicly available as open dataset.

I. Introduction

When deploying a service function chain (SFC) in a network function virtualization (NFV) scenario, resources like virtual CPU cores or memory have to be assigned to the involved virtualized network functions (VNFs) to meet performance goals like maximum delay or minimum throughput. These resources have usually been assigned manually based on expert knowledge, which is not feasible in agile environments in which softwarized networks are managed using DevOps methods [1]. In these environments, new (versions of) SFCs are automatically tested, continuously integrated, and deployed (CI/CD) [2] directly to production – a process that requires a priori knowledge about the relationship between assigned resources and achieved performance metrics to automate resource dimensioning and ensure that given performance goals are met [3]. This knowledge can be automatically gathered using VNF and SFC profiling (sometimes also called benchmarking) [4]–[7]. Such profiling processes collect information about the runtime behavior of the SFC under test (SFC-UT) by deploying it under different resource configurations and testing its resulting performance. The results of these profiling processes, the so-called SFC performance profiles (SFC-PP), are then used by management and orchestration (MANO) systems to optimize their resource dimensioning decisions. It is essential that the obtained SFC-PPs provide enough information to accurately describe the performance behavior of the deployed SFC to meet performance goals and to avoid unexpected performance degradations caused by, for example, automatically deployed service updates.

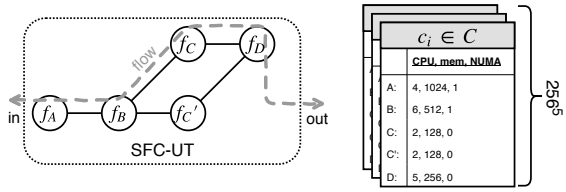

Fig. 1: Example scenario of an SFC with 5 VNFs and their configuration parameters

A challenge for such profiling solutions is, on the one hand, based on the fact that the configuration space sizes of even simple SFCs, which have to be explored during the profiling process, tend to become very large if the number of configuration parameters or number of involved VNFs increases. For example, a service with 5 VNFs, in which each VNF can have 1–16 CPU cores, $\{128, 256, 512, \ldots, 16384\}$ MB memory and non-uniform memory access (NUMA) enabled or disabled; this leads to $16 \cdot 8 \cdot 2 = 256$ possible configurations per VNF and 256^5 possible configurations for the entire SFC, as shown in Fig. 1. It can be seen that the configuration space of such an SFC is huge and it becomes infeasible to test each of these configurations during the profiling process – keep in mind that each test entails to re-deploy or re-configure the SFC-UT. On the other hand, the profiling processes as such are expected to be performed as part of the NFV DevOps cycle and thus have to be completed in a reasonably short time, i.e., a couple of hours or even minutes [1]. Because of this, novel profiling solutions are needed that do not require to exhaustively test the complete configuration space of an SFC. These profiling solutions aim to already produce usable SFC-PPs at the beginning of the profiling process and potentially improve their quality, in terms of profile accuracy, over time. In this context, profile accuracy means the error between the expected performance values described in the SFC-PP and the performance values achieved in reality.

In this paper, we introduce the concept of *time-constrained profiling (T-CP)* which produces SFC profiling results within a

978-3-903176-14-0

given time limit. Besides a a formal problem formulation given in Sec. II, the key contributions of this paper are twofold: First, we present the design, workflow, and used algorithms for a T-CP-enabled profiling system and introduce our open source T-CP prototype, called *nfv-t-cp* [8] in Sec. IV. Second, we use this prototype, which comes together with an open dataset containing the results of more than 2000 real-world SFC performance measurements, to evaluate our T-CP concept in Sec. V.

II. Problem Formulation

Given the huge configuration space of an SFC and the fact that the re-deployment or re-configuration of an SFC takes a considerable amount of time [9], executing profiling measurements on the complete configuration space is infeasible. It gets even worse if the profiling process should be performed in a given time frame, i.e., with a given time constraint l. As a result, only a subset of the complete configuration space can be tested and profiling measurements for untested configurations need to be predicted using the available results.

More formally, we define the set of all possible configurations F of a VNF as the cartesian product of a series of sets $F = P_1 \times P_2 \times ... \times P_m$ where each set represents a single, discrete configuration parameter (also called a *feature*) P_i and its possible values, e.g., number of CPU cores: $P_{\text{cores}} = \{1, 2, ..., 16\}$. Using discrete configuration parameters works fine since all relevant real-world configurations are also based on discrete values. In a complex SFC, multiple VNFs are combined and each of them can be configured independently of other VNFs with a configuration $c \in F$. For simplicity of the model, we assume that every involved VNF supports the same configuration space and thus all available configuration features of F. Based on this, the overall configuration space C of a complex SFC with n VNFs can be defined as $C = F_1 \times F_2 \times ... \times F_n$. Its cardinality as function of available configuration features and number of VNFs is given by $|C| = \left(\prod_{i=1}^{m} |P_i| \right)^n$.

For each SFC configuration that should be tested, the service is deployed and configured (taking setup time t_{up}), its performance is measured (in time t_{measure}), and it is terminated to free the resources (taking time t_{down}). After this, the next SFC configuration is tested. We call this a *profiling round* [5]. The total time t needed to profile a single SFC configuration is hence $t = t_{\text{up}} + t_{\text{measure}} + t_{\text{down}}$ where t is usually dominated by t_{up} and t_{down} [9]. In this paper, we consider t to be constant for each configuration to be tested.

As a result, the number of configurations k that can be tested in a given time limit l is limited by $k \leq \left\lfloor \frac{l}{t} \right\rfloor$. More specifically, we define the subset of configurations to be tested within the given time limit as $\tilde{C} \subset C$ and $|\tilde{C}| = k$. This subset is defined by a *selection function* $S_k : C \to \{0, 1\}$. For each configuration $c \in \tilde{C}$, we obtain profiling results, denoted by a function $\tilde{P} : \tilde{C} \to \mathbb{R}$ (profiling results could be tuples of real numbers or other values as well; this does not matter for the remaining discussion). We lift these *measured* profiling results, obtained on $\tilde{C}$, to *predicted* results for the entire configuration space by defining a profiling predictor $P : C \to \mathbb{R}$ that uses the measured results for $\tilde{C}$ as training data. We denote these predicted profiling results as $\hat{R}$.

This model poses two questions. First, how to best select the subset of k configurations to be tested? And second, using the performance measurements made on these k configurations, how to best pick the interpolation function P so that the resulting SFC-PPs accurately predict the performance of the SFC compared to real measurements, i.e., minimizing the prediction error? We will give answers to these two questions in the remainder of this paper.

To measure the quality of the predicted results $\hat{R}$ compared the measured results R, we use the *normalized root-mean-squared deviation (NRMSD)* as our main evaluation metric. The NRMSD is based on the mean-squared error (MSE), calculated over the entire configuration space with n predictions, and is normalized using the range ($R_{\text{max}} - R_{\text{min}}$) of the measured data (Eq. 1). We picked a normalized metric to ease comparison between SFC profiles with different scales.

$$\text{NRMSD} = \frac{\sqrt{\frac{1}{n} \sum_{i=1}^{n} (\hat{R}_i - R_i)^2}}{R_{\text{max}} - R_{\text{min}}} \tag{1}$$

III. Related Work

A couple of solutions for performance profiling of virtualized applications has already been proposed by the cloud-computing community. Most of them focus on solutions to profile single-VM applications [10], [11] but some solutions also support complex, composed applications [12]. In addition to this, the NFV community has also started to search for profiling solutions that focus specifically on NFV use cases [7], [13]. These solutions either focus on profiling single VNFs or on evaluating NFV infrastructure deployments [4], [14], [15]. Others do consider profiling of complex SFCs [16] to characterize the performance behavior of end-to-end services, which cannot be derived from isolated VNF profiles [5], [6]. Some of the SFC profiling solutions support even automated testing of different VNF sequences in an SFC [6]. All of these solutions, however, face the challenge of huge SFC configuration spaces that have to be explored, leading to impractical runtimes of the profiling process. None of the existing approaches can automatically select a subset of configurations to be profiled to systematically reduce the time needed to characterize an SFC. Even if it is possible to simply stop these existing NFV profiling solutions after the time limit is reached, irrespective of their current system state, it would cause very poor or even incomplete profiling results. This is because those solutions might only have explored a very small or unimportant part of the configuration space at the point in time when they are stopped. Because of this, smarter solutions are needed that start to profile the configuration space at large, in the first steps, and then successively improve the result quality until the time limit is reached. This is where our T-CP approach can be used as a complementary extension to the existing approaches as we do not tie our T-CP design to a specific profiling solution as shown in Sec. IV-A.

One solution that does smart selections in a cloud application context is called PANIC [17]. In their paper, the authors compare three selection approaches: Uniform grid, random sampling as well as their *greedy adaptive sampling algorithm (PGAS)*, each combined with different prediction (or interpolation) approaches, like linear regression. Their results indicate that testing a small subset of the overall configuration space already yields sufficient results to generate reasonably good performance profiles. Their evaluation focuses on a two-dimensional configuration space using cloud benchmarks based on big data frameworks like Hadoop. Even if their PGAS algorithm can be used for NFV scenarios, as we shown in Sec. V-B where we compare PGAS to the algorithms presented in this paper, the PANIC system as such was not designed for NFV scenarios and does not offer support for complex SFCs.

An extension to PANIC is a decision tree-based selection approach, which was recently proposed in [18]. This approach uses decision trees to iteratively partition the configuration space based on the accuracy of linear regression models applied to each of these partitions. The final partitioning and the selected configuration points obtained during the partitioning phase are then used to train a new decision tree-based model to finally represent the cloud application's performance behavior. According to [18], this approach tends to show a reduced accuracy when only small numbers of configurations are tested. Further, the approach tends to show better results when executed on configuration spaces with few, for example, two dimensions, which is rarely the case if SFCs, often consisting of more than two VNFs, should be profiled. As a result, their solution is not directly applicable to our scenarios. However, the use of decision trees seems to be a promising approach and we plan to explore it in future work.

Even though the presented solutions try to reduce the size of configuration spaces that have to be explored during profiling, none of them has a notion of time-constrained profiling nor supports the integration with NFV profiling platforms. This underlines the novelty of our T-CP concepts for the NFV domain, as presented in this paper.

IV. Designing a T-CP System

Based on our previous work in which we introduced a generic automation approach for VNF and SFC profiling [5], we designed an SFC profiling system that explicitly supports time-constrained profiling. The system gets all possible configuration parameters and a fixed time limit as inputs, runs automated performance measurements on the SFC-UT until the time limit is reached, and derives an SFC performance profile based on the available measurements.

A. Building Blocks and Workflow

Using the problem formulation (Sec. II) we identified and analyzed the required building blocks and workflows of a T-CP system and developed a prototype as shown in Fig. 2. Its components are placed around one or multiple profiling

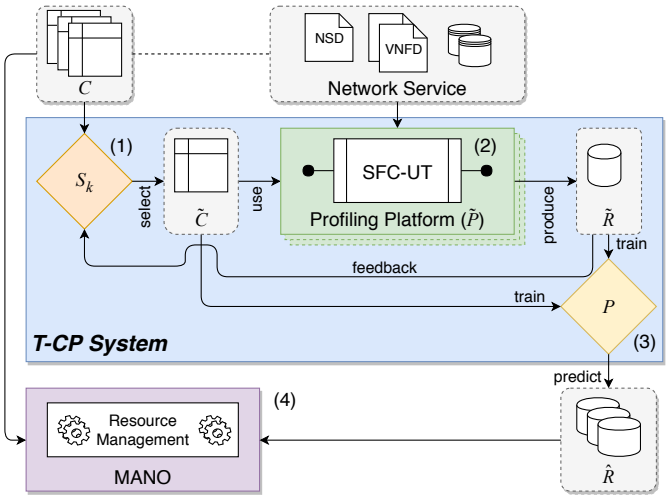

Fig. 2: Main building blocks and workflow of our T-CP system, build around existing profiling platforms, feeding the resource management component of a MANO system.

platforms that execute the SFC-UT and measure its performance under different resource configurations. Before that, the configurations to be tested are selected by the selection component S_k (step 1) and serve, together with the service description, as inputs to the profiling platform(s) (step 2). Note that we do not tie our T-CP system to a specific SFC profiling platform and designed it to use arbitrary, existing solutions. To combine other profiling platforms with T-CP, they need to have an interface where the configurations to be tested during the profiling runs can be specified as well as an interface to output the measured performance results – requirements that are fulfilled by [4]–[6]. It is important to note that the presented system design is not limited to a specific metric. Depending on the used profiling system, networking metrics like throughput and delay or even application-level metrics like number of video streams might be used. The measured results are forwarded to the prediction component P, which uses them as training data and generates predicted performance values for all possible configurations of the SFC-UT (step 3). Finally, the predicted results $\hat{R}$ can be forwarded to a MANO system to optimize its resource dimensioning decisions and for automated lifecycle actions, like scaling, healing, or reconfiguration (step 4). Alternatively, the obtained profiling results $\hat{R}$ can be used to analyze the behavior of the SFC-UT, e.g., by an SFC developer, to debug performance issues.

The two main building blocks that distinguish this system from existing profiling solutions [4]–[6] are the selection component S_k and the prediction component P. The selection component gets all possible configuration parameters of the SFC-UT (C) and a maximum number of k configurations to be selected as inputs before selecting the first configuration to be measured by the profiling platform. Once this is done, the selection algorithm selects the next configuration to be

measured. This round-based design allows to not only use static selection algorithms, like random sampling, but also dynamic selection algorithms that use feedback of already performed measurements as additional input, potentially improving the selection quality. We present an example of a feedback-based selection algorithm in the next section. If multiple parallel profiling platforms are available in a T-CP system, a centralized S_k component selects the configurations to be profiled which are then equally distributed among the available profiling platforms (see Sec.IV-B). The resulting measurements are collected and processed by a centralized prediction component as described in Sec. IV-C.

B. Selection Component

We implemented three different selection algorithms for our T-CP system. First, a simple random selection algorithm, called URS, that selects configurations uniformly at random and does not rely on the feedback of previous measurements.

Second, we re-implemented the PGAS selection algorithm proposed by Giannakopoulos *et al.* [17] as a first example for a feedback-based algorithm. PGAS initially picks a fixed number of samples at the borders of the configuration space (border points) before picking further samples based on the maximum distances between the measured results of the previous samples. PGAS was initially designed to profile cloud applications instead of NFV service chains but can be plugged into our proposed T-CP system.

Third, we developed our own feedback-based algorithm, called *weighted random VNF selection (WRVS)*, which comes in two flavors, WRVS1 and WRVS2. WRVS's general idea is to favor the configurations that belong to the VNFs of the SFC that seem to have a higher impact on the overall SFC performance. To detect those VNFs, the algorithm is split into two phases. First, a *bootstrapping phase* is used to select n (WRVS1) or $2n$ (WRVS2) initial configuration points for an SFC with n VNFs. Each of these configuration points minimizes or maximizes the configuration parameters of one of the n VNFs and sets the configuration parameters of all other VNFs to their median values. More specifically, WRVS1 picks configurations that minimize parameters and WRVS2 picks configurations that minimize and maximize the parameters. For example, in an SFC with three VNFs in which only the number of vCPU cores (1...16) can be configured, the first two configuration points, using WRVS2, would be `(vnf1=1, vnf2=8, vnf3=8)` and `(vnf1=16, vnf2=8, vnf3=8)`. Thus, *vnf1* is once tested with minimum number of vCPUs and once with the maximum number of vCPUs, with the goal to gain knowledge about the impact of the vCPU configuration parameter of *vnf1*. The definition on how to minimize and maximize parameters, is given by the developer of the VNFs or by the profiling experiment developer as annotation to the configuration space used as input to our system. Using this initial selection scheme, the algorithm collects information about the impact of the individual VNFs to the overall SFC performance. This impact is defined as the change to the overall SFC performance Δ_i when the configuration of VNF i is altered. These values are used as weights in the next phase of the algorithm.

The second phase of the algorithm starts after n or $2n$ configurations have been tested and randomly picks configurations from the overall configuration space until the limit of k configurations is reached. This random selection process uses the weights from the first phase to favor the selection of configuration points that alter the configurations of those VNFs that have higher weights assigned. That is, configuration points from VNFs with a higher Δ_i are more likely to be selected for further profiling rounds. With this, the feedback from the profiling process guides our selection algorithm to improve the overall profiling result by focusing on those parts of the configuration space that seem to have higher impact to the overall SFC performance.

C. Prediction Component

This component can either be based on simple regression techniques or on more complex machine learning solutions. In our prototype, we utilize the the *scikit-learn* machine learning library [19] to implement the prediction component. Besides a simple *polynomial regression predictor*, we also use *support vector regression* predictors with different kernels, *decision tree regression*, *lasso regression*, *elastic net regression*, *ridge regression*, and *stochastic gradient descent regressions* predictors, resulting in a total of 11 prediction algorithms available in our prototype, as we further detail in the next section.

V. Evaluation

We use our T-CP system prototype, called *nfv-t-cp* [8], to evaluate our design and to study the impact of different selection algorithms, prediction algorithms, service topologies, and number of samples on the overall result quality, i.e., prediction error. To do so, we execute a set of profiling experiments in which all possible configurations of the SFC-UT are tested. This exhaustive profiling step gives us baseline results serving as ground truth for later comparison. Then we use T-CP to execute profiling experiments that only test a fixed number of configurations k and compare their results to the initial experiments using the *NRMSD* metric. In all our experiments, we decided to use *maximum throughput* as VNF and SFC performance metric captured during the profiling measurements. However, our prototype offers support for arbitrary metrics as long as they can be captured by the used profiling system.

Our evaluation uses two different approaches to provide our T-CP system with profiling data. The first set of experiments does not rely on real-world measurements and simulates the SFC-UT with a model that randomly assigns synthetic performance functions, taken from [18], to the involved VNFs as described in Sec. V-A. The second set of experiments, in contrast, does rely on real-world measurements and is based on a forwarding SFC with three VNFs that was also used in our previous work [5] and is described in Sec. V-B.

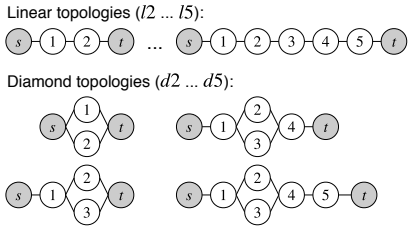

Fig. 3: Simulated SFC topologies

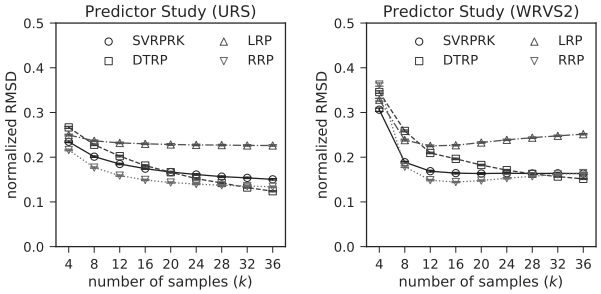

Fig. 4: Comparison of prediction algorithms

A. Randomized Synthetic Performance Models

In the first set of experiments, we simulate the performance behavior of an SFC-UT with a model that is based on Giannakopoulos *et al.* [18] and uses synthetic functions to map VNF configurations to performance (i.e. throughput) values. The benefit of this model-based approach is that our system can be evaluated with a high number of possible VNF and SFC configurations without the need to execute each of them in a real-world system and measuring their performance to get the required results to serve as ground truth.

For each evaluated T-CP system configuration, we generate 10 synthetic performance model instances and randomly assign one of the VNF performance functions taken from [18] to each of the VNFs. With this model, black-box SFCs with unknown VNF behavior are simulated.

We use eight different SFC topologies, composed of two to five VNFs, combined to four linear chains (l2 to l5) and four diamond chains (d2 to d5) as shown in Fig. 3. The performance of each VNF is calculated using a given configuration and assigned performance function. Since we selected throughput as metric for our evaluation, this results in an SFC graph with throughput values assigned to each node. Having this, we calculate the overall throughput of the SFC by solving the *maximum flow* problem between s and t assuming unlimited link capacities since we are only interested in modeling the VNF performance. Each VNF in this model has a single configuration parameter with 10 discrete configuration values, resulting in up to 10^5 SFC configurations for the largest topologies (l5 and d5).

We first analyze the behavior of different prediction algorithms, taken from [19], used in our T-CP system using the URS and WRVS2 selectors. Fig. 4 shows a comparison of four prediction algorithms using the following configurations:

- *Support vector regression (SVRPRK)* with RBF Kernel, $C = 1.0$ and $\epsilon = 0.1$.
- *Decision tree regression (DTRP)* with maximum tree depth set to 5.
- *Lasso regression (LRP) with* $\alpha = 0.1$.
- *Ridge regression (RRP) with* $\alpha = 0.1$.

Results for the other implemented prediction algorithms, which produce similar results, are not shown because of space limits. The figures show the NRMSD for different numbers of samples k selected from the overall configuration space of the profiled SFCs (up to 10^5 possible configurations). Each of these experiments is repeated 100 times and the error bars indicate $95\,\%$ confidence intervals. The results show that already a small number of samples, e.g., $k = 16$, allows to perform reasonably good predictions for the behavior of the overall configuration space of an SFC. They also show that the RRP and DTRP predictors perform best for the used SFCs.

Next, we analyze the behavior of different selection algorithms. This should answer the question whether our WRVS algorithm outperforms existing algorithms for cloud application profiling, namely PGAS [17]. Fig. 5 shows comparisons of four selection algorithms for each of the four prediction algorithms used before. It can be seen that the WRVS algorithms outperform the PGAS algorithm irrespective of the used prediction approach. However, the results also show that the WRVS selection is not better and often outperformed by the simple randomized selection approach (URS). In the SVRPRK case, WRVS shows slightly better results than URS for $8 \leq k \leq 20$, but is again outperformed by URS when k increases. This result is surprising since we expected that favoring some VNFs in the profiled SFC would lead to better results for small k.

We also compare the behavior of different selector/predictor combinations for different SFC topologies as shown in Fig. 6. The results show that the WRVS selectors show similar behavior for different topologies but tend to be better with smaller topologies, like *d2* and *d3*.

B. Real-world Performance Measurements

The second set of experiments utilizes our previous work about automated SFC profiling [5] and is based on a real-world profiling process using an SFC-UT with three VNFs. This SFC has a smaller configuration space but allows us to evaluate our system in a real-world scenario including realistic performance numbers that are based on measurements instead of synthetic functions. The used SFCs are linear chains that contain three real-world VNFs (*Nginx*[1] configured as TCP load balancer, the TCP relay *Socat*[2] and *Squid Proxy*[3] with disabled caching functionality to forward every packet) in different orders, resulting in three possible SFC topologies. Each VNF has a single configuration parameter (CPU time)

[1] http://nginx.org
[2] http://www.dest-unreach.org/socat/
[3] http://www.squid-cache.org

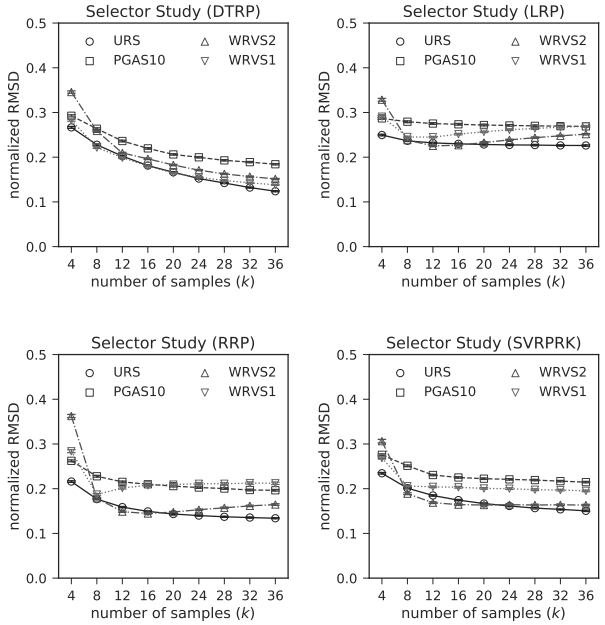

Fig. 5: Selector comparison with four prediction algorithms

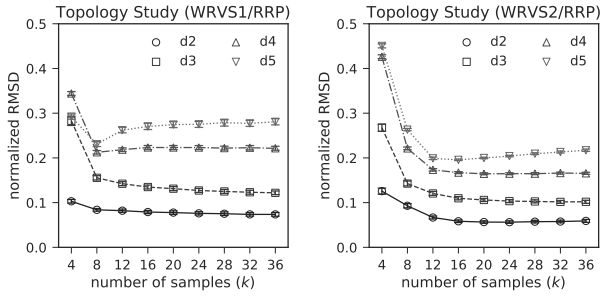

Fig. 6: Selector/predictor performance breakdown for different diamond SFC topologies with two to five VNFs

and the maximum achieved throughput during a large HTTP download is measured. The measurements to test each possible configuration of this simple chain took about 39 hours, with 60 seconds measurement time per configuration. Our raw measurements are available online and can be re-used by other researchers [8].

We again compare the behavior of our selection approaches in combination with four different prediction algorithms as shown in Fig. 7. The results show that the WRVS algorithm works better on the real-world SFC performance data with three VNFs compared to the synthetic SFCs analyzed in the last section. Especially for small k, WRVS often outperforms URS and PGAS. However, especially for $k = 4$ the models are often overfitted when WRVS is used. This could be a result of the static selection of the initial points during the bootstrapping phase of WRVS.

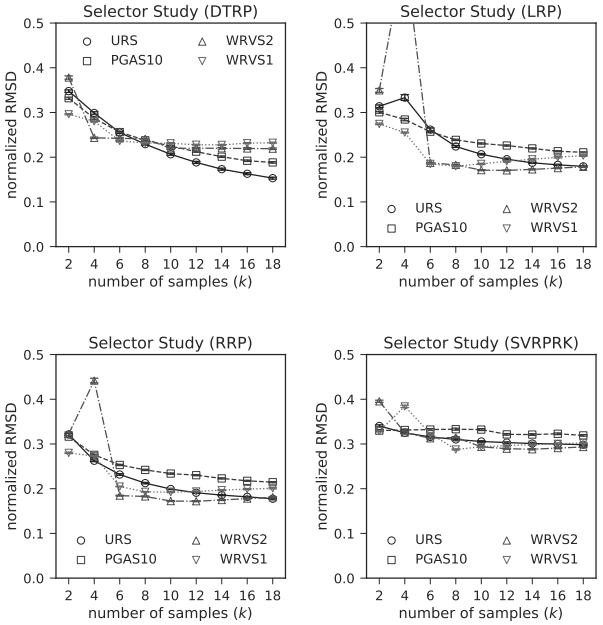

Fig. 7: Selector comparison using real-world measurements

VI. Conclusion

The sizes of real-world SFC configuration spaces makes the application of existing NFV profiling solutions infeasible for agile DevOps environments. Even though existing NFV profiling solutions could be simply stopped after a given amount of time, the produced performance profiles would only reflect a small subset of the configuration space and would loose important information about the SFC-UT's performance behavior. We presented a solution for this by introducing our T-CP concepts for NFV.

To study these concepts, we presented our open-source T-CP system *nfv-t-cp* [8] and used it to analyze different selection and prediction algorithms. Our results show that a T-CP system can generate reasonably accurate SFC-PPs by profiling only small subsets of the overall configuration space. We showed that the subset of configuration points that are profiled has a big impact to the quality, in terms of prediction error, of the resulting SFC-PPs. Our presented selection algorithm performs well with most of the real-world cases but barely keeps up with random selection approaches in the synthetic scenarios. This motivates future work on selection algorithms, for example, based on decision trees as shown in [18]. Together with this paper, we published an open dataset of our profiling experiments containing performance measurements of real-world forwarding SFCs with three VNFs [8].

Acknowledgments

This paper has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. H2020-ICT-2016-2 761493 (5GTANGO), and the German Research Foundation (DFG) within the Collaborative Research Centre "On-The-Fly Computing" (SFB 901).

REFERENCES

[1] H. Karl, S. Dräxler, M. Peuster, A. Galis, M. Bredel, A. Ramos, J. Martrat, M. S. Siddiqui, S. van Rossem, W. Tavernier *et al.*, "DevOps for network function virtualisation: an architectural approach," *Transactions on Emerging Telecommunications Technologies*, vol. 27, no. 9, pp. 1206–1215, 2016.

[2] M. Zhao, F. L. Gall, P. Cousin, R. Vilalta, R. Muñoz, S. Castro, M. Peuster, S. Schneider, M. Siapera, E. Kapassa, D. Kyriazis, P. Hasselmeyer, G. Xilouris, C. Tranoris, S. Denazis, and J. Martrat, "Verification and validation framework for 5g network services and apps," in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2017, pp. 321–326.

[3] M. Peuster and H. Karl, "Understand Your Chains: Towards Performance Profile-based Network Service Management," in *5th European Workshop on Software Defined Networks (EWSDN'16)*. IEEE, 2016.

[4] R. V. Rosa, C. Bertoldo, and C. E. Rothenberg, "Take your vnf to the gym: A testing framework for automated nfv performance benchmarking," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 110–117, 2017.

[5] M. Peuster and H. Karl, "Profile Your Chains, Not Functions: Automated Network Service Profiling in DevOps Environments," in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2017.

[6] J. Nam, J. Seo, and S. Shin, "Probius: Automated approach for vnf and service chain analysis in software-defined nfv," in *Proceedings of the Symposium on SDN Research*, ser. SOSR '18. New York, NY, USA: ACM, 2018, pp. 14:1–14:13. [Online]. Available: http://doi.acm.org/10.1145/3185467.3185495

[7] R. Rosa and C. Rothenberg, "VNF Benchmarking Methodology," IETF Internet-Draft https://tools.ietf.org/id/draft-rosa-bmwg-vnfbench-01.html, UNICMAP, Internet-Draft, 2018. [Online]. Available: https://tools.ietf.org/id/draft-rosa-bmwg-vnfbench-01.html

[8] M. Peuster, "nfv-t-cp: NFV Time-Constrained Profiling Framework," online at: https://github.com/CN-UPB/nfv-t-cp, Paderborn University, 2018.

[9] T. L. Nguyen and A. Lebre, "Virtual machine boot time model," in *2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, 2017, pp. 430–437.

[10] T. Wood, L. Cherkasova, K. Ozonat, and P. Shenoy, "Profiling and modeling resource usage of virtualized applications," in *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*. Springer-Verlag New York, Inc., 2008, pp. 366–387.

[11] J. Taheri, A. Y. Zomaya, and A. Kassler, "vmbbthrpred: A black-box throughput predictor for virtual machines in cloud environments," in *European Conference on Service-Oriented and Cloud Computing*. Springer, 2016, pp. 18–33.

[12] B. C. Tak, C. Tang, H. Huang, and L. Wang, "Pseudoapp: performance prediction for application migration to cloud," in *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*. IEEE, 2013, pp. 303–310.

[13] ETSI GS NFV-TST 001, "Network Functions Virtualization (NFV); Pre-deployment Testing; Report on Validation of NFV Environments and Services," 2016.

[14] R. V. Rosa, C. E. Rothenberg, and R. Szabo, "VBaaS: VNF benchmark-as-a-service," in *2015 Fourth European Workshop on Software Defined Networks*. IEEE, 2015, pp. 79–84.

[15] M. Baldi and A. Sapio, "A network function modeling approach for performance estimation," in *Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI), 2015 IEEE 1st International Forum on*. IEEE, 2015, pp. 527–533.

[16] L. Cao, P. Sharma, S. Fahmy, and V. Saxena, "NFV-VITAL: A Framework for Characterizing the Performance of Virtual Network Functions," in *Network Function Virtualization and Software Defined Network (NFV-SDN), 2015 IEEE Conference on*. IEEE, 2015, pp. 93–99.

[17] I. Giannakopoulos, D. Tsoumakos, N. Papailiou, and N. Koziris, "Panic: modeling application performance over virtualized resources," in *Cloud Engineering (IC2E), 2015 IEEE International Conference on*. IEEE, 2015, pp. 213–218.

[18] I. Giannakopoulos, D. Tsoumakos, and N. Koziris, "A decision tree based approach towards adaptive modeling of big data applications," in *2017 IEEE International Conference on Big Data (Big Data)*, 2017, pp. 163–172.

[19] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.