A Hardware-in-the-loop Wireless System Design: Benchmarking NFV Network Services

Dillon J. Horton, Thai T. Vu, Tu N. Nguyen, Kun Suo, Ahyoung Lee, Selena He, and Yong Shi
Department of Computer Science, Kennesaw State University, Marietta, GA 30060, USA.

***Abstract*—5G/6G network slicing provides a method to split network infrastructure into self-contained slices, each comprising various virtual network functions (VNFs) mapped onto physical nodes. Developing robust resource allocation algorithms to efficiently embed VNFs into the available nodes of a network has been a significant focus of research into network slicing. A key component of many of these mapping algorithms is the need for accurate measurements of the resource and bandwidth requirements of the VNFs. This problem is further compounded by the performance impacts of collocating multiple VNFs on one physical node. Competition between collocated VNFs for shared hardware resources can lead to performance degradation and an inability to meet their service level agreements (SLAs). To tackle this, we propose a novel VNF profiling framework for benchmarking a set of VNFs, accounting for the effect of collocation on their performance, namely Collocation-Aware Function Mapping (CAFM). Unlike prior research, which mostly relies on simulations, we conduct comprehensive experiments on a real-world deployment environment based on the POWDER platform to benchmark collocated VNFs. The experimental results demonstrate that the CAFM framework can accurately detect and mitigate collocation issues causing performance degradation, ensuring that VNFs can meet their SLAs more effectively.**

***Index Terms*—VNF, NFV, Network slicing, Benchmarking.**

I. Introduction

Network Slicing and VNFs. Network slicing is a 5G/6G technology that allows physical infrastructure to be logically split into multiple network slices [1]. A key component of implementing network slicing is Network Function Virtualization (NFV) architecture. NFV seeks to replace network functions traditionally implemented through special-purpose equipment with ones implemented as software running on general-purpose machines [2]. Each slice consists of a set of Virtual Network Functions (VNFs), which together provide a service or set of services. Each set of VNFs is embedded into the physical infrastructure based on their requirements and the available resources of the physical nodes as illustrated in Fig. 1. Prior research in 5G/6G networks has examined optimizing the performance of these network slices through *optimal mapping of VNFs* [3], [4], often relying on *measurements of computing resources* including CPU and memory usage, as well as *network resources* such as network traffic. To best optimize slice performance, these algorithms rely on the *accurate profiling of the resource requirements of each VNF* [5], [6].

We thank the anonymous reviewers for their suggestions and feedback. This research was in part supported by US NSF under Grants: CNS-2103405, AMPS-2229073, CPS-2103459, SHF-2210744, and CNS-2244450. Corresponding author: Tu N. Nguyen.

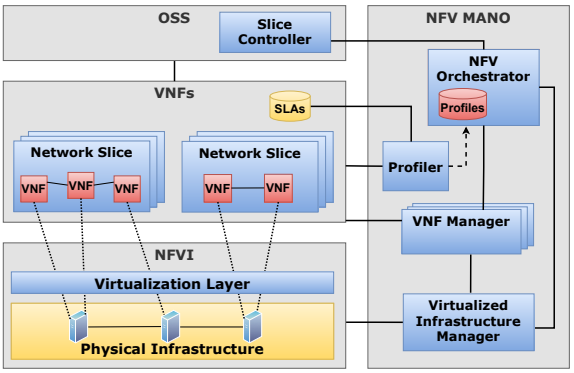


Fig. 1. Network slicing in an NFV architecture.

VNF Profiling. Prior work looked at collecting the measurement data through VNF profiling platforms [6], [7]. These systems *monitor* and *benchmark* VNFs, then use the data to generate VNF profiles. VNF profiles are defined by *computational models* which describe the performance characteristics and resource requirements of VNFs. These profiles can then be given to NFV Management and Orchestration systems (MANO) (Fig. 1), which use this data to create deployments of VNFs and manage resources to ensure that each VNF can meet their service level agreements (SLAs) [1], [8].

Collocation and Performance Degradation. Due to the limited number of physical machines, VNFs often need to be *collocated with other VNFs on the same machine*. This leads to challenges as collocated VNFs compete for hardware resources causing performance degradation which ultimately reduces Quality of Service (QoS) [8]. For this work, we use the packet throughput of each VNF as our QoS indicator. The impact of performance degradation can be seen in Fig. 2, which shows the normal throughput of each tested VNF (Fig. 2(a)) as well as the maximum throughput drop seen by each VNF during experiments where it was collocated (Fig. 2(b)). Throughput drop in this case is the percentage decrease in the throughput of a VNF when it is collocated, compared to the ideal when tested in isolation. To maintain the SLAs, NFV orchestration schemes need to address performance degradation caused by collocation. Although there exist methods mitigating the performance degradation due to collocation, they mostly rely on *prediction or isolation with limitations* [8], [9], [11]. Prediction algorithms often inaccurately estimate extra resource needs, and methods isolating collocated VNFs can be ineffective or lead to poor resource utilization. Furthermore, these works do not clearly define how prediction can be utilized in VNF mapping.

Our Contribution. To address the drawbacks of the above prediction and isolation methods, we propose a hybrid VNF mapping framework, namely *Collocation-Aware Function Mapping (CAFM)*, which incorporates and improves upon current performance prediction algorithms in order to efficiently create optimized VNF deployment configurations. We make the following contributions:

- We implement a performance prediction methodology based on prior work which uses a multivariate contention vector to predict VNF performance. We test the prediction in a real NFV cloud environment in order to ensure accuracy of predictions.
- We propose a novel VNF mapping algorithm, which incorporates performance prediction in order to minimize collocation induced performance drops. Additionally, we combine VNF performance prediction and mapping contributions to form our novel framework, *CAFM*.
- We implement an example use case experiment, which evaluates the proposed methodology's impact on mapping effectiveness and final configuration performance. Unlike previous simulation based work, we take steps to ensure the experiment accurately represents the real-world environment these VNFs would be deployed in. We compare the final performance and efficiency to standard VNF mapping techniques.

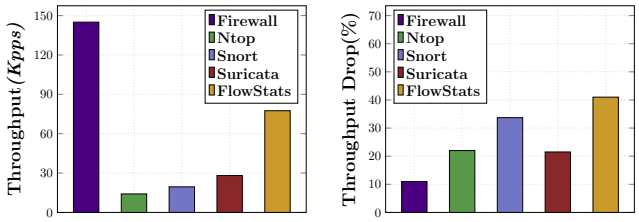
Organization. The rest of this paper is organized as follows. Section II provides a background on VNF mapping algorithms, VNF benchmarking, performance prediction, motivation, and the problem. The system model and solution are presented in Section III. Section IV describes the system design and implementation. The experiment and evaluation are provided in Section V. Finally, Section VI concludes this paper.

II. Preliminary

In this section, we provide background information on NFV architecture and the algorithms for mapping VNFs onto physical infrastructure. Additionally, we conduct an in-depth review of related work on performance benchmarking and prediction.

A. Background

NFV Architecture. Consider the NFV architecture depicted in Fig. 1, comprising four primary subsystems: (1) *NFV Management and Orchestration (NFV MANO)*, (2) *NFV Infrastructure (NFVI)*, (3) *Virtualized Network Functions (VNFs)*, and (4) *Operation Support System (OSS)*. The NFV MANO (Management and Orchestration) system manages the life-cycle of network slices and the VNFs comprising them, determining the deployment of VNFs and the allocation of resources from the underlying NFVI. The NFVI consists of pools of hardware resources abstracted and logically partitioned by a software virtualization layer. The VNF component encompasses all VNFs to be deployed, along with relevant information about them, such as the requirements set by their SLAs. Lastly, the OSS (Operations Support System) is responsible for initiating the creation of new slices and provides all the information



(a) Normal throughput. (b) Max observed throughput drop.

Fig. 2. Comparison of throughput and throughput drop of tested VNFs.

stored in the VNF component. Our proposed framework fits within the NFV MANO component of the NFV architecture and seeks to facilitate the effective use of profiling and performance prediction data in deployment decisions.

VNF Mapping Algorithms. NFV MANO systems rely on VNF mapping algorithms to efficiently and effectively map VNFs onto physical infrastructure [3], [4]. These algorithms are diverse and employ various methods and data to determine service placement. A popular approach involves using the computational and network resource requirements of a VNF to decide its placement [5]. However, much of the current literature overlooks the impact of collocation and resource contention on the final performance of a VNF. While a few works attempt to account for collocation, they often use ineffective performance prediction models and fail to properly integrate them with VNF mapping algorithms.

Sources of Contention. To accurately measure shared resource contention, it is important to understand which resources VNFs are competing for. In NFV architectures, it is common practice for VNFs to be deployed on dedicated cores to help prevent CPU-related resource contention. Thus, the main sources of contention are the resources shared among cores. We focus on three sources of memory contention [8], [10], [15].

- **Last Level Cache (LLC)**. While each core has dedicated L1 and L2 caches, the LLC is shared between cores and thus is a source of contention for VNFs.
- **Packet I/O**. Many commonly used Intel architectures implement Data Direct I/O (DDIO) as part of optimizations for packet I/O. DDIO isolates a portion of the LLC to be dedicated for packets being processed by the system. Contention arises when the number of packets exceeds the available space in the cache.
- **Main Memory**. Lastly, collocated VNFs compete for the limited memory bandwidth of the shared system.

B. Motivation

Benchmarking VNFs to determine their requirements for deployment using these algorithms is generally done in two ways: using *Isolated benchmarking* and using *performance prediction*. Below, we examine each approach.

VNF Benchmarking Tools. Some prior work exists focusing on benchmarking VNFs [13], [14]. Rosa et al. developed a VNF and NFV benchmarking framework called

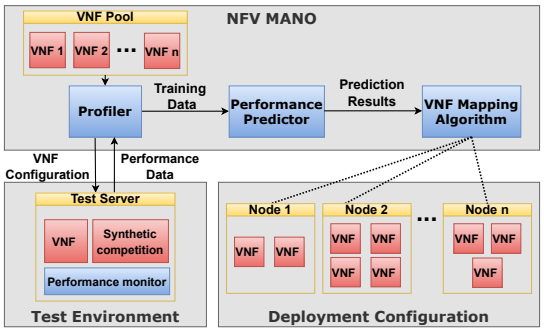


Fig. 3. System design of proposed framework (CAFM).

"gym", which focused on automating the benchmarking process for VNFs [13]. Other approaches employed machine learning techniques to verify the results of performance predictions [14]. However, these systems were designed without considering the impact of collocation on VNF performance, and instead, they focused on these VNFs in isolation. As such, their benchmarking data could lead to an ultimately inaccurate portrayal of a service's performance once it is deployed. One way to prevent collocation from affecting the VNF benchmarks is to try the isolation of deployed VNFs. This can be accomplished by leveraging Intel's Memory Bandwidth Allocation and Cache Allocation Technology (CAT) to try and fully isolate the memory and cache resources of collocated VNFs [11]. However, this often fails to fully isolate resources and previous work has found that isolation can lead to inefficient resource utilization [8].

Performance Prediction. A key aspect of performance prediction is its ability to predict the impact of collocation on performance for multiple VNFs. The concepts of a VNF's *contentiousness* and *sensitivity* are critical to accurate performance prediction of collocated VNFs [8], [9], [15]. Contentiousness is a measure of how much a VNF will impact the performance of collocated VNFs through the usage of shared hardware resources. Conversely, sensitivity is a measure of how much a VNF will be impacted by collocated contentious VNFs. The metrics used to measure contentiousness and sensitivity can vary between algorithms but they consist of one or more metrics associated with shared resources, such as cache or I/O buffer usage [8], [9].

Early work focused on predicting contention through a single metric. BubbleUp measures the contention through the working set size of the VNF [9]. Dobrescu et al. is another such work and instead measures contention through the cache access rate [10]. However, only measuring a single metric gives an incomplete picture of resource contention, which can impact the accuracy of the prediction. When measuring the accuracy of these prediction models on newer hardware, they were found to have some prediction errors as high as 70% [8]. Manousis et al. improved upon these earlier works by considering several contention-related metrics to determine the sensitivity and impact of each VNF. [8]. Although this led to a significant improvement in prediction accuracy, higher error rates were still observed for larger numbers of collocated services. The presence of prediction errors means service providers using this data need to be more conservative in resource allocation to ensure compliance with service level agreements (SLAs). This could lead to under-utilization and inefficient usage of the allocated resources.

III. System Design and Proposed Solution

As a part of the NFV architecture (Fig. 1), our proposed *CAFM* framework consists of three main components as depicted in Fig. 3, which are as follows: the profiler, the performance prediction algorithm, and the VNF mapping algorithm. We expand on the work of [8] to build or profiler and performance prediction model. We additionally propose a novel heuristic VNF mapping algorithm (LPPD) for applying the generated prediction models to the VNF mapping problem. We explain each of the components in depth below.

A. Profiler

The goal of the VNF profiler is to generate *profiles* which describe resource usage characteristics and SLA constraints of the VNF. The profiler takes as input a set of VNFs and their expected network traffic level and outputs a profile for each VNF. The profiler gathers data on the CPU and memory requirements of the VNF. In order to provide the necessary data for the prediction algorithm, the profiler also needs to determine the contentiousness and sensitivity of each VNF. Each profile is generated offline prior to the runtime deployment of each VNF. Exactly what metrics are collected during profiling and how they are used is dependent upon the chosen performance prediction algorithm. For profiling and performance prediction, we use an improved version of the prediction algorithm proposed in [8]. We modify the algorithm by reducing the number of metrics composing the contention vector of our server architecture and using mixed packet size traffic in order to better simulate real deployment scenarios. We find the models are just as accurate while being generated more quickly due the reduced number of metrics (Section §V).

TABLE I
Collected PCM metrics and the sources of contention they are related to.

PCM Metric	Description	LLC	Packet I/O	Memory
IPC	Num. Instructions		✓	
L3MISS	LLC miss rate	✓		✓
L3HIT	LLC hit rate	✓		
L2MISS	Cache access rate	✓		
READ	Memory read		✓	✓
WRITE	Memory write		✓	✓

Measuring Contentiousness. Contentiousness is the amount of pressure a VNF applies to the shared resources of the node it is deployed on. Resource contention occurs in three main areas, the LLC, Packet I/O, and main memory. Prior work has shown that using a single metric is not sufficient to accurately measure contention [8], [10]. As such, we select a series of hardware metrics to characterize the contentiousness of a VNF in these three areas. We collect

these metrics using the Intel PCM framework, an API designed to collect data from the hardware's performance monitoring units (PMUs) [16]. The collected metrics are shown in Table I, along with the sources of contention they are intended to be representative of. Together, these metrics form a vector that measures the contentiousness of a VNF. It is important to note that the contentiousness of a VNF also changes in the presence of competition (i.e., a VNF will have higher contentiousness when collocated with several other VNFs). As such, we implement the contentiousness composition method proposed in [8]. The profiler profiles the target VNF when running solo, as well as when collocated with a variable number of synthetic VNFs. When the performance predictor needs to know the contentiousness vector of a VNF, it first checks how many VNFs it will be collocated with and then takes the average contention vector from the profiler experiments with that number of collocated VNFs.

Measuring Sensitivity. Sensitivity is a measure of how a VNF reacts to the presence of competition for shared resources. To characterize sensitivity, we seek to create a model which takes a VNF and the contentiousness vector of its collocated competitors as input and outputs the predicted performance of the VNF when collocated with that competition. In order to build this model, we use the contentiousness vectors and collected performance data from our real experiments. This requires testing a large set of collocation configurations which are representative of the competition the VNF might have when deployed. To this end, we develop a synthetic competitor VNF, which can apply a variable amount of stress on the shared resources of the node. The synthetic competitor implementation is discussed in Section IV. The profiler collects VNF throughput data under various competitor configurations and contention vectors, used as training data for our gradient boosting regression model. Each VNF's unique response to resource contention requires individual sensitivity models built before deployment.

B. Performance Predictor

The performance prediction algorithm is built largely off the sensitivity models generated by the profiler. Under some scenarios, this is quite simple. For instance, given two VNFs, F_a and F_b, where we need to predict the expected performance drop of F_a when collocated with F_b, we simply plug the contentiousness vector of V_b into the sensitivity model for F_a. However, adding a third collocated VNF F_c complicates the prediction. In order to predict the performance of F_a when collocated with both other VNFs, the prediction algorithm first combines the two contentiousness vectors V_b and V_b to form the vector V_c^b. This can be done by taking the sum or average of each metric in the component vectors depending on the metric. The vector V_c^b can then be plugged into the sensitivity model to get the predicted performance drop of F_a.

C. VNF Mapping

Once the sensitivity models for each VNF have been trained, they can be passed to the VNF mapping algorithm, along with the set of physical infrastructure nodes they can be deployed on. We propose a **Lowest Predicted Performance Drop (LPPD)** algorithm (Algorithm 1), which incorporates the prediction models to ensure high QoS under collocated scenarios. For each VNF, the algorithm takes a greedy approach to find the node with the lowest average predicted performance drop and maps the VNF to that node, updating the available resources based on VNF resource. The algorithm continues until all VNFs have been mapped to a node and outputs the set of mappings as the configuration to be used during deployment. It is important to note that predicting performance drop involves both the performance drop of the VNF being mapped as well as all VNFs currently on the node that would be affected by the addition of a new collocated VNF.

Algorithm 1: Lowest Predicted Performance Drop

Input : A set of K VNFs $\mathbb{F} = \{F_i\}$ and a set of P physical nodes $\mathbb{N} = \{N_j\}$
Output: A set of mappings $\mathbb{M} = \{M_i\}$ from VNFs to nodes

```
begin
    for (i = 1; i ≤ K; i = i + 1) do
        D_min ← 999        ▷ Initialize with a large number
        for (j = 1; i ≤ P; j = j + 1) do
            if N_j has the minimum required CPU and
              memory resources for F_i then
                D_i^j ← Predict(F_i, N_j).
                if D_i^j < D_min then
                    D_min ← D_i^j; M_i ← mapping(F_i, N_j).
    Return M
```

IV. Experiment Design and Implementation

In this section, we first design a POWDER-testbed-based NFV system. We then implement the proposed CAFM framework and configure tools for collecting and analyzing experimental results.

A. Experiment Design

1) **POWDER Testbed.** We utilize POWDER [12] to compare the performance of the NFV services in an environment similar to a real-world deployment. The POWDER testbed is a flexible infrastructure that enables a wide range of software defined experiments.
2) **OpenStack.** In order to better emulate real-world NFV architectures, we run our experiments using OpenStack, a commonly used cloud-computing platform [19]. Using POWDER, we configure an OpenStack deployment with one dedicated control node and one or more compute nodes. Each node has a 2.4 GHz, 64-bit Intel Quad Core Xeon E5530 processor and 12GB of RAM [12].
3) **Service Selection.** We test five widely used network services: Snort, Suricata, Ntop, Firewall (Click), and Flowstats (Click). These services were chosen to provide a wide range of resource usage in order to generate varied contentiousness vectors and sensitivity models.

B. Performance Prediction Experiment

Offline profiler. Here, we discuss our implementation of the offline profiler. The goal is to collect a representative set of VNF performance data which can train an accurate sensitivity model for performance prediction. This data is collected through accessing the hardware's PMUs. The exact PMUs available are hardware-dependent. The hardware specifications such as cache size, will also impact the results of offline profiling. Consequently, the profiler takes as input a VNF, its configuration, the expected traffic profile, and the server architecture of the cluster it will be deployed on. For our offline profiling scenario, we use the control node and one compute node configured through OpenStack. We implement and test five VNFs through the following software.

- **Snort.** Snort is an open-source intrusion detection system (IDS). We run Snort configured with the community rule set of roughly 400 rules [17].
- **Suricata.** Suricata is an open-source network analysis and threat detection software. We run Suricata in IDS mode, configured with the Suricata 5.0.10 rule set [18].
- **Click.** Click is a modular router designed to run various packet processing/routing tasks [20]. We configure Click to implement two VNFs. The first is a simple stateless firewall with roughly 1000 sequential rules. The second is a program to provide flow statistics on the captured traffic.
- **Ntop.** Ntop is an open-source traffic and security network traffic monitoring software. We configure Ntop to perform deep packet inspection on test network traffic [21].

Additionally, we implement a synthetic competitor VNF to cover a wide range of contentiousness vectors. The synthetic competitor is a configurable VNF, which can stress each of the shared resources to a variable degree. We generate various configurations of the synthetic VNF in order to fully cover the contentiousness vector space. When under test each VNF is deployed as a VM on the compute node along with a variable number of synthetic competitors. We test each VNF with real traffic captured and replayed using the Tcpreplay tool.

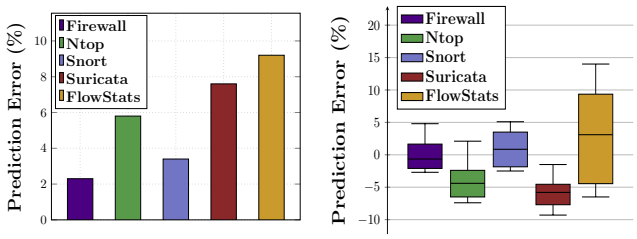
C. VNF Mapping Experiments

Unlike prior work, which relies on simulations, we conduct experiments on a real-world deployment environment for a use case of the CAFM framework. Using the POWDER, we run the experiments with one control node and 20 compute nodes, set up through OpenStack. We run our proposed LPPD algorithm and continually provide random VNFs from our set of implemented VNFs to minimize performance degradation. Additionally, we compare it to the performance of a traditional resource-based algorithm which does not account for collocation running in the same scenario [5]. This algorithm sorts the nodes by available resources and deploys the VNF to the node with the most available resources. The algorithm is tested with a variable number of VNFs from 10 to 80.

V. Experimental Evaluation

We analyze the performance of experiments running on the POWDER. We first examine results from the profiler's contention data collection and discuss the challenges of modeling VNF sensitivity. We then discuss the accuracy of our model in predicting the collocation-induced performance drop of a VNF. Finally, we examine a use case where the *CAFM* framework is used to deploy a set of VNFs in a real cloud environment.

A. Prediction Model

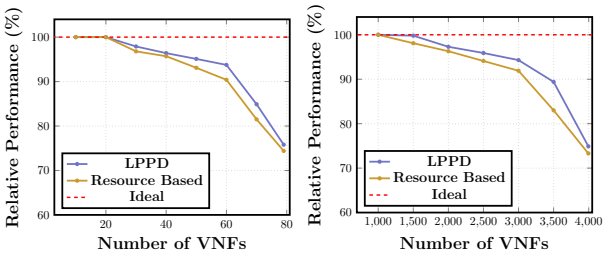


(a) Average prediction error. (b) Signed Prediction Error.

Fig. 4. Prediction error for VNF performance drop predictions.

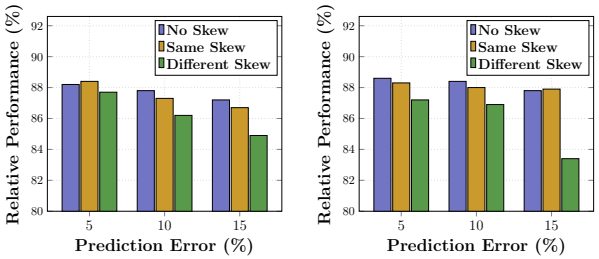
Prediction Accuracy. Before testing our mapping algorithm, we need to ensure the accuracy of our chosen prediction model. Fig. 4 shows the overall prediction error for five tested VNFs. From Fig. 4(a), we find that our prediction model is accurate, with an average prediction error of 5.6%. Fig. 4(b) shows the signed prediction error where a negative value represents the predicted performance drop being lower than the actual and a positive value represents the predicted performance drop being higher than the actual. The model does not appear to show a significant skew for under- or over-prediction across VNFs. However, this does not mean that the model does not produce skewed prediction models for individual VNFs. For instance, the model seems to consistently underpredict the performance drop of Suricata and Ntop. This could lead to potential SLA violations when deployed. We can help reduce this error by continuing to collect performance data after deployment and using it to slowly update and improve the model overtime. A natural concern is the possibility of greater performance degradation due to the overhead of deploying a performance monitoring tool. However, the performance monitoring tool does not need to measure the contentiousness of the VNFs deployed on the node, as this data would have already been collected during profiling. It only needs to collect the throughput data to see the real performance drop. This new collected data can be fed back to the profiler, allowing the sensitivity model of each VNF to be periodically updated over the lifetime of deployment. This helps improve the overall accuracy of the model as new VNFs are added to the network.

VNF Mapping Experiment. Fig. 5(a) shows the results from our VNF mapping experiment on the real system. We find the mappings generated by our collocation-aware algorithm (LPPD) consistently outperform the mappings generated by the purely resource based algorithm with an average throughput of 91.8% compared to an average throughput of 90.1%.



(a) Real experiment (20 nodes). (b) Large scale experiment (1000 nodes).

Fig. 5. The relative performance of deployed VMs compared to the ideal of them running in isolation.



(a) 20 nodes and 70 VNFs. (b) 1000 nodes and 3500 VNFs.

Fig. 6. Demonstration of how VNF performance changes with prediction error skew. Deployments we generated using LPPD.

Due to the limited size of the real system, we additionally test the algorithm in a simulated environment with 1000 nodes to verify the usability of the algorithm in large scale cloud networks, the results of which are shown in Fig. 5(b). The performance improvements are greater in the large scale experiment with the LPPD showing an average throughput of 93.1% and the resource based algorithm showing an average throughput of 90.4%. The benefit of the LPPD algorithm is greatest when the number of VNFs is between 60% and 90% of the maximum capacity of the network. Deployments with a low number of VNFs will be running VNFs in isolation or collocated with only a single other VNF with low resource contention leading to prediction becoming less important.

We observe that the impact of prediction error is dependent on the skew of the prediction error. To examine this, we run the LPPD algorithm with only two VNF types. We artificially modify the prediction error of two chosen VNFs and measure the impact on overall performance for different skews and levels of error (Fig. 6). We find that prediction error has a much greater impact on mapping algorithm when two VNFs in the pool have opposite skews. This is due to the fact that opposite skews push the algorithm to heavily favor the performance of one VNF over the other, causing it to mistakenly ignore overall more efficient deployments.

VI. Conclusion

Developing robust resource allocation algorithms to efficiently embed VNFs into the available nodes of a network has been a significant focus of research into network slicing. We have proposed the CAFM framework for combining performance prediction and VNF mapping. The framework consists of a profiler for benchmarking VNFs, a performance predictor for predicting the impact of collocation and a VNF mapping algorithm (LPPD) for using the prediction model to generate VNF deployments minimizing collocation-induced performance drop. We have successfully implemented our framework in a real cloud environment using POWDER and OpenStack, demonstrating how VNFs are deployed in a real-world setting. The numerical results show that the LPPD algorithm can reduce the total performance drop of the system compared to traditional VNF mapping methods.

References

[1] S. Zhang, "An overview of network slicing for 5g," *IEEE Wireless Commun.*, vol. 26, no. 3, pp. 111–117, 2019.

[2] B. Chatras, *et al.*, "Nfv enabling network slicing for 5g," in *IEEE ICIN 2017*, 2017, Conference Proceedings, pp. 219–225.

[3] K. J. Ambarani, *et al.*, "Enforcing resource allocation and vnf embedding in ran slicing," in *IEEE GLOBECOM*, 2021, Conference Proceedings.

[4] C. Hernández-Chulde, *et al.*, "Vnf placement over autonomic elastic optical network via deep reinforcement learning," in *IEEE ICC*, 2023, Conference Proceedings, pp. 422–427.

[5] W. Attaoui, *et al.*, "Vnf and cnf placement in 5g: Recent advances and future trends," *IEEE Trans. Netw. Service Manag.*, 2023.

[6] N. Ferdosian, *et al.*, "Profile-based data-driven approach to analyse virtualised network functions performance," in *IEEE ISCIT 2023*, 2023, Conference Proceedings, pp. 306–311.

[7] S. Moazzeni, *et al.*, "A novel autonomous profiling method for the next-generation nfv orchestrators," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 1, pp. 642–655, 2021.

[8] A. Manousis, *et al.*, "Contention-aware performance prediction for virtualized network functions," in *ACM SIGCOMM*, 2020.

[9] J. Mars, *et al.*, "Bubble-up: Increasing utilization in modern warehouse scale computers via sensible co-locations," in *IEEE/ACM MICRO*, 2011.

[10] Dobrescu, Mihai, *et al.*, "Toward predictable performance in software Packet-Processing platforms," 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12), pp. 141-154, 2012

[11] V. R. Chintapalli, *et al.*, "Nfvpermit: Toward ensuring performance isolation in nfv-based systems," *IEEE Trans. Netw. Service Manag.*, vol. 20, no. 2, pp. 1717–1732, 2023.

[12] J. Breen *et al.*, "Powder: Platform for open wireless data-driven experimental research," in *WiNTECH*, 2020.

[13] R. V. Rosa, *et al.*, "Take your vnf to the gym: A testing framework for automated nfv performance benchmarking," *IEEE Commun. Mag.*, 2017.

[14] X. Vasilakos, *et al.*, "ion-profiler: Intelligent online multi-objective vnf profiling with reinforcement learning," *IEEE Trans. Netw. Service Manag.*, 2024.

[15] L. Tang, *et al.*, "Contentiousness vs. sensitivity: improving contention aware runtime systems on multicore architectures," *ACM EXADAPT*, 2011, Conference Proceedings, pp. 12-21

[16] Intel, "Intel Performance Counter Monitor," [Online]. Available: https://github.com/intel/pcm. [Accessed: June 20, 2024].

[17] Snort, "Snort: Network Intrusion Detection & Prevention System," [Online]. Available: https://www.snort.org. [Accessed: June 20, 2024].

[18] Suricata, "Suricata: Open Source Network Threat Detection Engine," [Online]. Available: https://suricata.io/. [Accessed: June 20, 2024].

[19] OpenStack, "OpenStack: Open Source Cloud Computing Software," [Online]. Available: https://www.openstack.org/. [Accessed: June 20, 2024].

[20] E. Kohler, *et al.*, "The click modular router," *ACM Trans. Comput. Syst.*, vol. 18, no. 3, pp. 263–297, 2000

[21] Ntop, "Ntopng," [Online]. Available: https://github.com/ntop/ntopng. [Accessed: June 20, 2024].