This work has been accepted for publication in 2018 IEEE Conference on Network Softwarization (Netsoft 2018).
 ISBN: 978-1-5386-4633-5

Generating Resource and Performance Models for Service Function Chains: The Video Streaming Case

Sevil Dräxler, Manuel Peuster, Marvin Illian, Holger Karl
Paderborn University, Paderborn, Germany
{sevil.draexler, manuel.peuster, millian, holger.karl}@uni-paderborn.de

***Abstract*—Understanding the behavior of the components of service function chains (SFCs) in different load situations is important for efficient and automatic management and orchestration of services. For this purpose and for practical research in network function virtualization in general, there is a great need for benchmarks and experimental data. In this paper, we describe our experiments for characterizing the relationship between resource demands of virtual network functions (VNFs) and the expected performance of the SFC, considering the individual performance of the VNFs as well as the interdependencies among VNFs within the SFC. We have designed our experiments focusing on video streaming, an important application in this context. We present examples of models for predicting the interdependence between resource demands and performance characteristics of SFCs using support vector regression and polynomial regression models. We also show practical evidence from our experiments that VNFs need to be benchmarked in their final chain setup, rather than individually, to capture important interdependencies that affect their performance. The data gathered from our experiments is publicly available.**

I. Introduction

Service function chains (SFCs) for services like Internet video streaming consist of several virtual network functions (VNFs) like encoders, caches, and content servers. The SFCs are deployed on top of an infrastructure, typically managed by an orchestration system that places, deploys, and scales the SFC and its VNFs. For example, to ensure that the users are served with an acceptable latency, the orchestration system must instantiate the right number of instances for each VNFs in the right location in the underlying network. The right amount of resources needs to be calculated and allocated to each VNF instance so that the fluctuating amount of load can be handled without violating any service-level agreements or exceeding the capacity constraints of the underlying network. To do so, the orchestration system needs knowledge about the SFC, typically provided by the *descriptors* of the VNFs and the SFC.

Existing descriptors for VNFs rely on the knowledge of the component developer to provide an estimation of the amount of resources required to handle a given load with a target performance level. The VNFs can then be deployed with the requested amount of resources and scaled out/in, e.g., upon reaching pre-defined thresholds. We argue that this approach has two serious shortcomings, that we highlight using our experiments: (1) Resource demands of a VNF depends on the load and the targeted performance (more details in Section IV). Therefore, defining a fixed and constant set of resources to be allocated to each VNF can result in over-/under-estimating the required resources and lead to sub-optimal states for both the service and the underlying network. (2) Resource demands and performance of a VNF in an SFC depends on the allocated resources and the performance of other VNFs in an SFC (more details in Section V). Therefore, any attempt to model the resource demands of a VNF that is chained together with other VNFs needs to consider the dependencies to other VNFs, as well as the dependencies to the amount of load at each point in time.

Having correct models to predict the resource demands of the VNFs in an SFC, it is possible to optimize the service life-cycle management operations, e.g., for optimal and dynamic placement and scaling of the VNFs. In previous work [1], we have proposed a flexible *service template embedding* approach for optimizing the scaling and placement of SFCs in a single decision step. In this approach, a *service template* describes the required components of a network service and the desired connection patterns among them. Additionally, the template specifies the resource demands of each VNF as a function of the load it needs to handle. The load can be characterized, for example, in terms of the data rate on each incoming connection point of the service component. Using such a service template and based on the current load, the components are scaled out and embedded into the network.

Predicting the relationships between the resource demands of VNFs (e.g., CPU, memory) and the targeted values of performance metrics of interest for each SFC (e.g., frame rate, video resolution) is cumbersome to do for a developer. Taking the interdependencies among VNFs into account makes it even more complicated to model and predict these relationships manually. Therefore, an automatic profiling step needs to be integrated into management and operation systems to identify and characterize these relationships. We have proposed such profiling approaches in previous work [2], [3]. Such an automatic profiling system could be validated by testing it against well-known services with different performance characteristics. However, there is a pervasive lack of such benchmarks in network function virtualization (NFV) – essentially no experimental data for virtual network functions (VNFs) is publicly available. To overcome this lack, we have set up a testbed to characterize such relationships in a video streaming scenario, which is a common application deployed on distributed networks. We present some results from analyzing the data from these experiments. The data

is publicly available [4]. We hope that this might be a first step towards creating NFV benchmarks, to ensure that further research can take place on a solid empirical foundation.

After an overview of existing work for profiling service components (Section II), we describe our experimental setup (Section III) and introduce our prediction models based on polynomial regression and support vector regression (Section IV). Finally, we describe examples of interrelationships we have observed among resource demands and performance of different VNFs in an SFC (Section V) before concluding the paper.

II. Related Work

In this section, we give an overview of related approaches to the work we present in this paper.

Gmach et al. [5] determine the minimum resource needs to run a certain workload, based on the distribution of historic traces in a data center setup. Rasoolzadeh et al. [6] focus on optimizing energy consumption by estimating and adjusting the CPU cycles required for video encoding. Fan and Wang [7] compare responsiveness of web servers under a heavy request load, in a bare-metal environment. In contrast to these attempts, our work focuses on characterizing resource requirements to fulfill a certain target performance.

Xu et al. [8] model the relationship between throughput and response time of a web server, for request rates below the peak load. In their experiments, they vary the memory allocation and the number of CPU time slices before potentially being preempted. Do et al. [9] focus on predicting the performance of applications on VMs. Given a target performance, their models determine whether it can be achieved on a certain system. Giannakopoulos et al. [10] approximate service performance, given a certain hardware configuration. They use neural networks and linear regression and approximate the performance using a Gaussian distribution. We also approximate and verify functions that take available resources as input and yield possible values for certain performance metrics. Moreover, we use support vector regression (SVR) as a machine learning technique and polynomial regression (PR).

In previous work [2], we have analyzed the practical requirements of a profiling system to generate performance behavior information, which can be used to support resource allocation decisions for VNFs in an SFC. Similar approaches have been presented for profiling individual components in the cloud computing context [11] and in the NFV context [12]. In a follow-up work [3], we have proposed an automated system for profiling complete SFCs. Our observations in our experiments in the current paper confirm and complement our previous findings regarding the necessity of benchmarking VNFs in their chained setup rather individual profiling of the VNFs without considering the interdependencies within an SFC.

III. Measurement Methodology

We performed a large set of performance measurements using a real-world SFC to collect the initial data needed to build and train realistic performance models. We have done the measurements with individually deployed VNFs as well as with a fully deployed SFC. This gave us interesting insights about the deviating performance behavior of some VNFs when they are executed as part of a chain compared to individual deployment. Sec. V presents more details about the observed effects. The data from all our experiments is available for public use [4]; in this paper, we only show a subset of results collected from a video encoding and buffering SFC.

TABLE I
Parameters and values used for data collection

Parameter	Values
#vCPUs encoder	1 to 4
#vCPUs cache	1
Codecs	H.264, H.265
Videos	bunny [14], docu1 [15], docu2 [16], game [17], noise [18]
Resolutions	426x240, 640x360, 854x480, 1280x720, 1920x1080
Frame rates	24, 30, 40, 50, 60
Target bit rates	1000 Kb/s to 22000 Kb/s

Fig. 1 shows our measurement setup that is based on an OpenStack Octa testbed running running on four physical machines with Intel(R) Core(TM) i5-4690 CPU @ 3.50GHz and 16GB memory. We have configured three of these machines as compute nodes so that each VNF and the simulated user VM can be executed on its own physical machine, to eliminate noisy neighbor effects [13] from our measurements. All machines are interconnected with two 1 GigE links, one for control and one for the data plane.

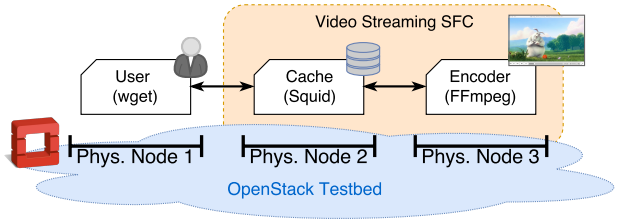


Fig. 1. Measurement testbed with the used video streaming SFC (*Cache* ↔ *Encoder*) and the simulated users running in three VMs deployed on three physical compute nodes.

The results shown in this paper are based on a video streaming SFC consisting of a video encoder VNF (*FFserver*[1] and *FFmpeg*[2]) and a cache VNF (*Squid 3.5.12*[3]) configured and built with default settings and installed in Ubuntu 16.04 VMs. To simulate users that access the video streams, we used the HTTP client *wget*[4] deployed in an additional VM as shown in Fig. 1.

During our experiments, the users access a video stream delivered by the cache VNF and encoded by the encoder VNF on-the-fly. For each new experiment, the cache was restarted

[1]https://ffmpeg.org/ffserver.html
[2]https://ffmpeg.org/ commit 148c8e8
[3]http://www.squid-cache.org
[4]https://www.gnu.org/software/wget/

so that all streaming content had to be fetched from the encoder instead of using cached streaming data. We collected the CPU and memory utilization of each of the used VNFs, transmission statistics (like data rates) between encoder and cache, as well as between cache and users. We also recorded application-level metrics, like encoded frames per second. For each run, we configured the encoder VNF to compress a 60-second video, given in a resolution of 1920x1080, to a target resolution between 426x240 and 1920x1080, using target bit rates between 1000 Kb/s and 22000 Kb/s, frame rates between 24 and 60, and using either the H.264 or H.265 encoding standard. The used encoder requires a preset value for the trade-off between video quality and encoding time, which we set to *medium* for videos with low bit rates and good visual quality. Table I summarizes the full list of used parameters for the executed measurements. We have conducted a total of 18500 experiments with these configurations.

IV. Analysis

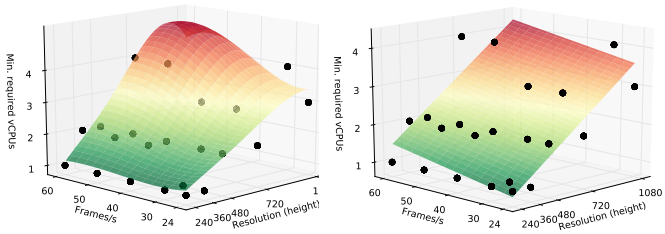
In this section, we present examples of the prediction models we have developed. An extended description of some of these models and more detailed descriptions of the analysis are also available in [19].

We used models based on SVR and PR for predicting the minimum required vCPUs for the encoder. The training data we have used for creating the SVR-based model consists of the test runs where the frame rate was never below the targeted frame rate, using the minimum number of vCPUs among all such observations. We have set C=10^6 and ϵ=10^{-12} after testing a wide range of values for these parameters and evaluating them based on mean squared error (MSE) and their visual representations. Figure 2(a) shows a plot for predicting the number of vCPUs based on bit rate, resolution, and frame rate. To be able to visualize this 4-dimensional relationship, we have fixed the bit rate to 5500 Kb/s in this plot. All plots shown in the rest of the paper are generated based on the data from experiments with the H.264 encoding standard.

For the PR-based approach, we have tested degrees 0 to 10 for the polynomial. While degree 7 gave the lowest mean squared error, the plots suggest a highly over-fitted model using this degree. Figure 2(b) shows the PR model (for bit rate 5500 Kb/s) using the following 1st-degree polynomial, which resulted in the best trade-off between MSE value and observable over-fitting. $c(b, r, f)$ represents the number of vCPUs, given bit rate b, resolution r, and frame rate f[5]. Black dots represent the measured data points.

$$c(b, r, f) = 3.29 \cdot r + 1.77 \cdot f + 1.10 \cdot b - 0.96$$

[5]All coefficients in this and all following functions have been rounded to 2 decimal points. b is given as Kb/s, r as height of the video in pixels assuming a 16:9 aspect ratio, and f as frames/s. Moreover, we have divided the values of these parameters by powers of 10 in our experiments, such that all values are between 0 and 1, to avoid computational errors we were observing in our SVR models using the actual values.



(a) SVR approach (b) PR approach

Fig. 2. Prediction of required vCPUs for encoder based on bit rate, resolution, and frame rate, shown for bit rate of 5500 Kb/s.

Comparing the mean squared errors and different plots of the SVR and PR models, the SVR-based approach gives better predictions for the minimum number of required vCPUs.

We have also developed additional SVR and PR models to predict the minimum number of required vCPUs c based on the target bit rate b, resolution r, or frame rate f individually. We omit the corresponding plots due to space constraints and present only the corresponding functions as follows.

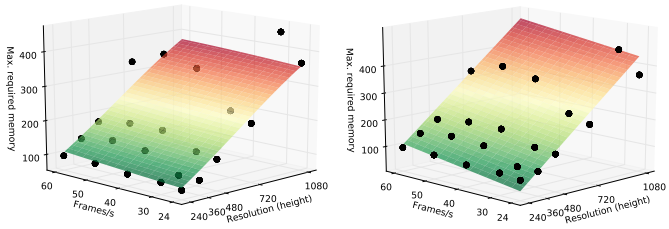
$$\begin{aligned} c(b) &= 15.12 \cdot b^5 - 50.31 \cdot b^4 + 62.12 \cdot b^3 - 35.20 \cdot b^2 \\ &\quad + 9.58 \cdot b + 0.10 \\ c(f) &= -0.017 \cdot f^2 + 0.93 \cdot f + 1.70 \\ c(r) &= -111.57 \cdot r^2 + 35.55 \cdot r + 0.73 \end{aligned}$$

Predicting and allocating the right amount of memory to the encoder ensures that no data is swapped into the hard drive and eliminates this performance-limiting factor for the video streaming application. Similar to the prediction models for the number of required vCPUs, we have developed SVR and PR models for predicting the maximum amount of required memory.

As training data for the SVR-based prediction model, we have taken the used memory in test runs where the minimum number of vCPUs are used and no violation of the target values for bit rate, resolution, and frame rate has occurred for all video files. Figure 3(a) shows the SVR-based model with C=10 and ϵ=10, for bit rate of 5500 Kb/s. With these values we could observe the lowest MSE. Figure 3(b) shows the PR-based model using a 1st-degree polynomial, which resulted in the best trade-off between the MSE and the over-fitting detectable in different plots. In this function, $m(b, r, f)$ represents the maximum required memory (in MB) to achieve a given bit rate b, resolution r, and frame rate f.

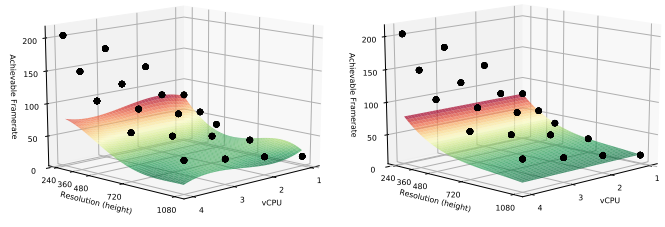
$$m(b, r, f) = 472.54 \cdot r + 196.77 \cdot f + 18.38 \cdot b - 130.51$$

The SVR-based model predicts the maximum required memory with a lower mean squared error than the PR-based model. However, examining different plots for both models shows that the predictions from both models cover the measured data points similarly well in most cases.



(a) SVR approach (b) PR approach

Fig. 3. Prediction of required memory based on bit rate, resolution, and frame rate, shown for bit rate of 5500 Kb/s.



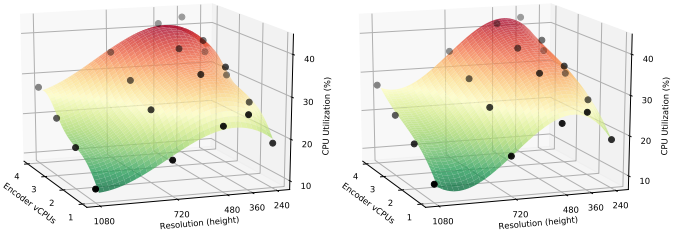
(a) SVR approach (b) PR approach

Fig. 4. Prediction of achievable frame rate based on bit rate, resolution, and number of vCPUs assigned to the encoder, shown for bit rate of 5500 Kb/s.

Using similar approaches, we have developed prediction models for the achievable frame rate based on bit rate, resolution, and the number of available vCPUs for the encoder. In our test runs, we have observed seconds where no frames were encoded, e.g., because of encoding delays. To prevent having these values distorting the prediction model, we have calculated the frame rate per configuration with a confidence interval at 95% confidence. For this, we have taken all values for frame rate $f(r, b, c)$ over all encoding processes using all values for resolution (r), bit rate (b), number of vCPUs (c), and all video files. Instead of the actual minimum value for observed frame rate, we have used the lower bound of the confidence interval as the minimum observed frame rate for training our model.

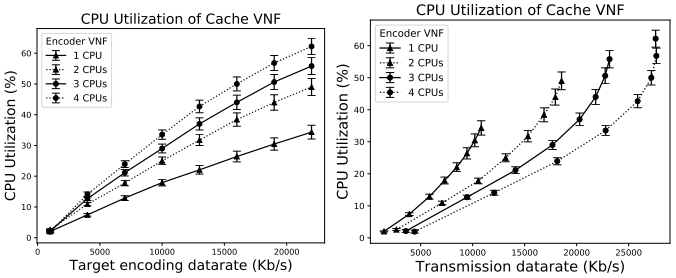
Figure 4(a) shows a plot for the SVR-based prediction, using $C = 10^8$ and $\epsilon = 10^{-7}$, for bit rate 5500 Kb/s. Figure 4(b) shows the PR-based prediction using a 4th-degree polynomial function. The function has 36 coefficients, so we omit it in the paper. As shown in the plots, both approaches give only partially reasonable predictions and cannot adapt to all values. The reason for this could be the large diversity in the observed values for different metrics. The model might be improved by analyzing different subsets of the gathered data individually, for example, by separating the data gathered with different number of vCPUs.

To capture the interdependencies among both VNFs in our SFC, we have also developed models to predict the CPU utilization of the cache VM based on the resolution and the number of vCPU cores that are allocated to the encoder VM. Figure 5(a) shows the SVR model, using $C = 10^2$ and $\epsilon = 10^{-2}$. Figure 5(a) shows the PR-based prediction using a



(a) SVR approach (b) PR approach

Fig. 5. Prediction of CPU utilization of the cache based on resolution and the number of vCPU cores assigned to the encoder.



(a) Target encoding data rate (b) Transmission data rate

Fig. 6. Avg. CPU utilization of cache VNF (Squid) over target encoding data rate and transmission data rate between encoder and cache VNF.

3rd-degree polynomial function, that resulted in the best MSE. It can be observed that the CPU utilization of the cache is clearly influenced by the allocated CPU to the encoder. In Section V we describe this interrelationship in more detail.

V. Unexpected Performance Interrelationships

We noticed some interesting effects during our measurements which indicate that performance data of VNFs should always be collected from setups where the VNFs are also executed as part of the full SFC they belong to. The reason for this is that VNFs might show different performance behavior for the same resource configuration when they are deployed as part of an SFC. Fig. 6 highlights such effects by showing the average CPU utilization of the cache VNF for different data rates and different configurations of the encoder VNF. More specifically, Fig. 6(a) shows the cache VNF's CPU utilization for different target bit rate settings at the encoder VNF and Fig. 6(b) shows values of the same metric as a function of the achieved transmission data rate on the link between encoder VNF and cache VNF.

Note that the CPU usage of the cache VNF changes when the number of vCPU cores assigned to the encoder VNF is changed. This can obviously not be a noisy neighbor effect since both VNFs are executed on independent physical machines. This means, the configuration of the encoder VNF has an effect on the performance of the cache VNF even though they are deployed in isolation. This result is a clear indicator for the need for SFC-based measurements as discussed in our previous work on SFC benchmarking [3], where we discovered

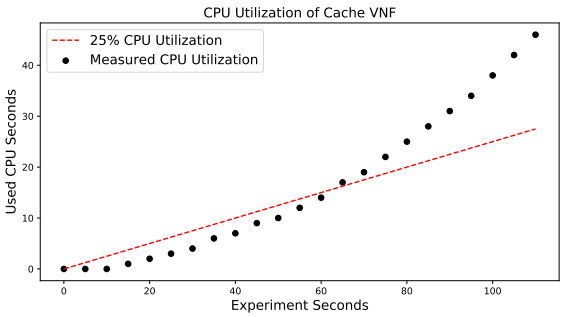


Fig. 7. Development of cache (Squid) CPU usage during a single experiment with constant incoming datarate

performance interference in SFCs with simple forwarding elements.

We believe that the effect is caused by the implementation of the cache VNF (*Squid*) in the presented scenario. The observations show that *Squid* requires more CPU time when the number of vCPU cores assigned to the encoder VNF is increased. With more cores assigned, the encoding performance of *FFmpeg* increases and the video stream is encoded faster, i.e., output data is produced with a higher rate by the encoder VNF and sent to cache VNF. As a result, *Squid* has to cache more bits per second which increases its CPU utilization. In addition to this, we observed that the CPU usage of *Squid* also increases with the amount of data it has already processed. Fig. 7 highlights this effect by showing the used CPU time of *Squid* as a function of experiment time in which data was cached with a constant rate. The figure clearly indicates that the CPU usage of *Squid* required for caching increases faster than data arrives. This is an effect we call *"VNF over-heating"*, which can only be observed, and thus reflected in the derived models, if the measurements are based on full SFC deployments. We plan further investigation of such chain-based interference effects as future work.

VI. Conclusion and Future Work

This work is an attempt to provide some insights into the performance and resource demands of components of a video streaming service. We have done experiments using a video encoding and streaming function, as well as a cache function under different resource constraints and different performance targets. Our models show the feasibility of characterizing the resource demands and performance metric values of VNFs. Moreover, our results clearly show that VNFs have to be profiled in the SFC setup, as they are planned to be executed. Only in this way, the resource utilization and performance dependencies among them can be captured and used for accurate prediction models. Our analysis shows that these relationships are non-trivial even for simple functions, reinforcing the need for experimental data for benchmarking and further analysis.

As our models are based on data from specific hardware settings, a more generic model can be obtained by normalizing the results based on experiments in different environments.

Acknowledgments

This work has been performed in the framework of the SONATA project, funded by the European Commission under Grant number 671517 through the Horizon 2020 and 5G-PPP programs. This work is partially supported by the German Research Foundation (DFG) within the Collaborative Research Center On-The-Fly Computing (SFB 901).

References

[1] S. Dräxler, H. Karl, and Z. Á. Mann, "Joint optimization of scaling and placement of virtual network services," in *17th IEEE/ACM Int. Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, 2017.

[2] M. Peuster and H. Karl, "Understand your chains: Towards performance profile-based network service management," in *5th IEEE European Workshop on Software Defined Networks (EWSDN)*, 2016.

[3] ——, "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)*, Nov 2017.

[4] "Data from the experiments in this paper." [Online]. Available: https://uni-paderborn.sciebo.de/index.php/s/G9q2hmUNg4n8LEg

[5] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper, "Capacity management and demand prediction for next generation data centers," in *IEEE Int. Conf. on Web Services*, 2007.

[6] S. Rasoolzadeh, M. Saedpanah, and M. R. Hashemi, "Estimating application workload using hardware performance counters in real-time video encoding," in *7th Int. Symposium on Telecommunications (IST)*, 2014.

[7] Q. Fan and Q. Wang, "Performance comparison of web servers with different architectures: A case study using high concurrency workload," in *3rd IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*, 2015.

[8] X. Xu, T. Xu, Y. Yin, and J. Wan, "Performance evaluation model of web servers based on response time," in *IEEE Conf. Anthology*, 2013.

[9] A. V. Do, J. Chen, C. Wang, Y. C. Lee, A. Y. Zomaya, and B. B. Zhou, "Profiling applications for virtual machine placement in clouds," in *4th IEEE Int. Conf. on Cloud Computing*, 2011.

[10] I. Giannakopoulos, D. Tsoumakos, N. Papailiou, and N. Koziris, "Panic: Modeling application performance over virtualized resources," in *IEEE Int. Conf. on Cloud Engineering*, 2015.

[11] J. Taheri, A. Y. Zomaya, and A. Kassler, *vmBBThrPred: A Black-Box Throughput Predictor for Virtual Machines in Cloud Environments*. Cham: Springer International Publishing, 2016, pp. 18–33. [Online]. Available: https://doi.org/10.1007/978-3-319-44482-6_2

[12] L. Cao, P. Sharma, S. Fahmy, and V. Saxena, "Nfv-vital: A framework for characterizing the performance of virtual network functions," in *2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*, Nov 2015, pp. 93–99.

[13] S. Govindan, J. Liu, A. Kansal, and A. Sivasubramaniam, "Cuanta: quantifying effects of shared on-chip resource interference for consolidated virtual machines," in *Proceedings of the 2nd ACM Symposium on Cloud Computing*. ACM, 2011, p. 22.

[14] "Big buck bunny." [Online]. Available: https://peach.blender.org/download/

[15] "TPB AFK: The pirate bay away from keyboard." [Online]. Available: http://www.imdb.com/title/tt2608732/

[16] "The art of playing." [Online]. Available: http://www.imdb.com/title/tt3996164/

[17] "Best plays of: The international 5 movie dota 2 compilation highlights." [Online]. Available: https://www.youtube.com/watch?v=ERTIWNfx93w

[18] "TV static noise HD 1080p." [Online]. Available: https://www.youtube.com/watch?v=DH0BQtwEAsMw

[19] M. Illian, "Prediction of resource requirements and performance of virtualised network functions in a video streaming context," Bachelor's Thesis, Paderborn University, 2017.