

Quantitative Analysis of Dynamically Provisioned Heterogeneous Network Services

Hadi Razzaghi Kouchaksaraei, Holger Karl
Computer Network Group
Paderborn University, Paderborn, Germany
{hadi.razzaghi, holger.karl}@uni-paderborn.de

Abstract—Services in Network Function Virtualization (NFV) can have a variety of requirements such as data rates, latencies, and cost that can change during the lifecycle of services. To meet these requirements, various hardware and software resources are suggested for implementing Virtualized Network Functions (VNFs). However, meeting all service requirements using one implementation option is not always possible. For example, to improve the performance of VNFs, using acceleration hardware is proposed. Although acceleration hardware can improve the performance of a network function, as they are expensive appliances, they increase the cost of services; this might not be desirable for a particular service user or load that can be handled by cheaper resources. Dynamically provisioning services can solve this problem in which different implementations of VNFs are switched on the fly as service requirements change. In this paper, we analyse this service provisioning approach in terms of performance, cost, and management overhead by experimenting an example VNF.

I. INTRODUCTION

Network Function Virtualization (NFV) services can have a variety of requirements that can change during the lifecycle of services. These requirements can be, for example, different data rates, latencies, and cost. To meet these requirements, various hardware resources or software structures are suggested [1] [2] [3]. Usually, however, there are trade-offs between different hardware resources or software structures to realise a service, and there is no one single solution that can meet all requirements. For example, to improve the performance of compute- and network-intensive NFs, using acceleration hardware such as Graphics Processing Unit (GPU) or Field-Programmable Gate Array (FPGA) is proposed [4] [1].

Although acceleration hardware can improve the performance of network functions, as they are expensive appliances, they increase the cost of services; this might not be desirable for a particular service user or load that can be handled by cheaper resources. This problem can be solved by dynamically provisioning network services: we switch to a different service implementation on the fly as service requirements change. This can provide service performance most suitable to the given requirements. For example, consider two versions of a Deep Packet Inspection (DPI) NF: v1 is a CPU-based, v2 an FPGA-based implementation. On one hand, deploying v1 would perform well and be cheap when the input data rate is low but it would not perform well for high data rates. On the other hand, v2 performs better at high data rate but is too expensive at low data rates [1]. To make a trade-off between

these two DPI versions, a service package consisting of both versions can be used that allows us to deploy the right version and then switch between them on the fly as the input data rate changes. We call this type of services *multi-version services*, a concept that is proposed in our previous work [5]. While the idea looks promising in theory, an experimental evaluation of such a service provisioning approach is still missing.

In this paper, we contribute to the dynamic provisioning of network services that are realised on heterogeneous resources based on different requirements. To this end, we analyze this approach in terms of performance, cost, and management overhead. We considered a virtual transcoder (vTC) as a case study and implemented a COTS-based and a GPU-assisted virtual transcoder. We evaluated these vTC versions by metrics such as resource utilization and processing time. Based on this evaluation, we analysed the cost of vTCs running on cloud-based COTS and acceleration resources. We, further, evaluated the management overhead of dynamic service deployment by measuring the deployment time of vTCs.

The rest of the paper is structured as follows. In Section II, we review related work. Dynamic service provisioning is evaluated in Section III, and, finally, we conclude the paper in Section IV.

II. RELATED WORK

Studies in the context of dynamic service provisioning and multi-version services can be categorised into (i) studies related to evaluating Virtualized Network Functions (VNFs) realised with different implementation options and (ii) studies related to analysing the dynamic deployment of multi-version services. The former category supports the idea of the multi-version services as it can prove that there are trade-offs between using different implementation options of VNFs. This validates the idea of having multi-version services in a system as without such trade-offs, there would not be any need to have multiple versions of a service, but simply scaling a single version of service up and down would be all that is needed for optimal performance. The latter category analyses the overhead generated by dynamically service provisioning which is more associated with management and orchestration of these services.

There are multiple studies (e.g., [6], [7], and [1]) in the first category. Nobach et al. [1] studied the performance and cost of COTS-based DPI versus FPGA-based DPI. In this work,

they implemented a DPI that uses COTS resources for all its tasks and another one that offloads network-intensive tasks onto FPGA resources. They have evaluated the throughput and latency of these DPIs having the size of input packets as a parameter. The results of their evaluation show a clear trade-off between performance in terms of throughput and latency and costs. Using FPGA-based DPI for packet sizes up to 512 bytes can improve the throughput up to 124 times. Also, it can improve the latency up to 6 times. However, in the case that packet sizes are bigger than 512 bytes, the performance improvement becomes negligible. With that, we can observe that the use of FPGA-based DPI is not a proper solution for large packets as it costs 2.3 times more than COTS-based DPI without having a significant performance improvement.

For the second category, however, there have not been many studies. In our previous work [5], we have evaluated dynamically deployment of multi-version services using simulation. However, an experimental evaluation of multi-version services along with cost and management overhead analysis is missing.

III. QUANTITATIVE ANALYSIS

We have conducted an experimental evaluation to analyse dynamically provisioning multi-version services. In this evaluation, virtualized Transcoder (vTC) has been used as a case study as it consists of compute-intensive processes that can be offloaded to acceleration hardware. A transcoder provides functionalities such as converting video encoding format and spatial resolution, video transposing, and video transcoding. These functionalities are provided to adjust the original video to the viewer's network datarate, device resolution, frame rate and so on. Transcoder is used in both Video On Demand (VOD) (e.g., Youtube, Netflix) and live-streaming services to provide high-quality video streaming experience for the viewers [8]. Employing vTC as an example, we have analysed performance, cost, and management overhead of multi-version services, which is discussed in the following.

A. Performance Analysis

To analyse the performance, we have implemented two versions of a vTC: (v1) a COTS-based vTC that is designed to only utilise CPU for video processing and (v2) a GPU-assisted vTC that offloads compute-intensive processes to GPU. Both versions are based on FFmpeg¹, which is a software-based transcoder providing a wide range of transcoding functionalities. To emulate the NFV environment, we have deployed vTCs on KVM-based virtual machines. The test-bed, that is used for the evaluation, consists of four VMs: VM #1 hosts a packet generator that breaks down videos to Group of Picture (GOP) [8] and embeds them into Real-time Transport Protocol (RTP) packet payloads to be sent to vTCs, VM #2 runs the COTS-based vTC, VM #3 hosts the GPU-assisted vTC, VM #4 receives the transcoded videos from vTCs and display them using a video player. All VMs were running on a server equipped with an Intel(R) Xeon(R) W-2123 CPU at 3.60GHz

(8 Processors), 8 GB DDR4 RAM, and an Nvidia GeForce RTX 2080 GPU.

We have considered two metrics namely the video transcoding processing time for the entire video and CPU/GPU utilization. These two metrics are considered as processing time can affect the total latency of video streaming services and CPU/GPU utilization is an indicator of the service cost. Parameters of the evaluation are the video bitrate and resolution. For a range of video resolutions, we have experimented the performance of vTCs. For each video resolution, we ran the test for a range of input bitrates. Both transcoders convert the incoming video format to H.264 format. Also, big buck Bunny² video have been used as the input video.

The video processing time results, illustrated in Fig. 1, show that the GPU-assisted vTC processes the videos much faster than the COTS-based vTC when the video has a high resolution and bitrate. Fig. 1e shows that the processing time difference can reach up to 40 seconds for videos with 1080p resolution and 1.6 MB/s Bitrate, which is indeed a remarkable difference. However, the processing time difference decreases as the video resolution gets lower. For example, looking at processing times for videos with 240p resolution (Fig. 1b), we see that the processing time difference goes down up to 1 second and, for 120p resolution videos (Fig. 1a), the COTS-based vTC even outperforms the GPU-assisted vTC. This is because, in the case of GPU-assisted vTC, there is an extra CPU and GPU communication overhead that does not exist in the COTS-based vTC. This increases the total processing time of the GPU-assisted vTCs; however, as this overhead is very low, it has no significant impact on the processing time of high-resolution videos.

The CPU utilization evaluation results are depicted in Fig. 2. As expected, the trend is the same as what we got in the video processing time evaluation. While the GPU-assisted vTCs utilizes less CPU for *high-resolution videos* (Fig. 2e), COTS-based vTC uses less CPU to process *low-resolution videos*. This is because there is no processing associated with exchanging data between CPU and GPU in the COTS-based vTC. Looking at the results, the CPU utilization of GPU-assisted vTC remains between 30% to 50% for all video resolutions, however, COTS-based vTC utilization varies from 20% for the 120p resolution (Fig. 2a) to 250% for 1080p resolution (Fig. 2e). 250% CPU utilization means 3 CPU cores are used for the process in which 2 of them are fully utilized and the other is 50% utilized.

We have also measured the GPU memory utilization of GPU-assisted vTC. The results are shown in Fig. 3. Performing this evaluation, we observed that changing the input bitrate does not change the GPU utilization and it always remains constant. However, the GPU utilization increases as the input video resolution gets larger.

Based on the processing time and resource usage evaluation, we observe that there is a trade-off between using COTS-based and GPU-assisted vTCs. Although using GPUs can improve

¹<https://ffmpeg.org/>, accessed May 18, 2019

²<http://bbb3d.renderfarming.net/download.html>, accessed May 27, 2019

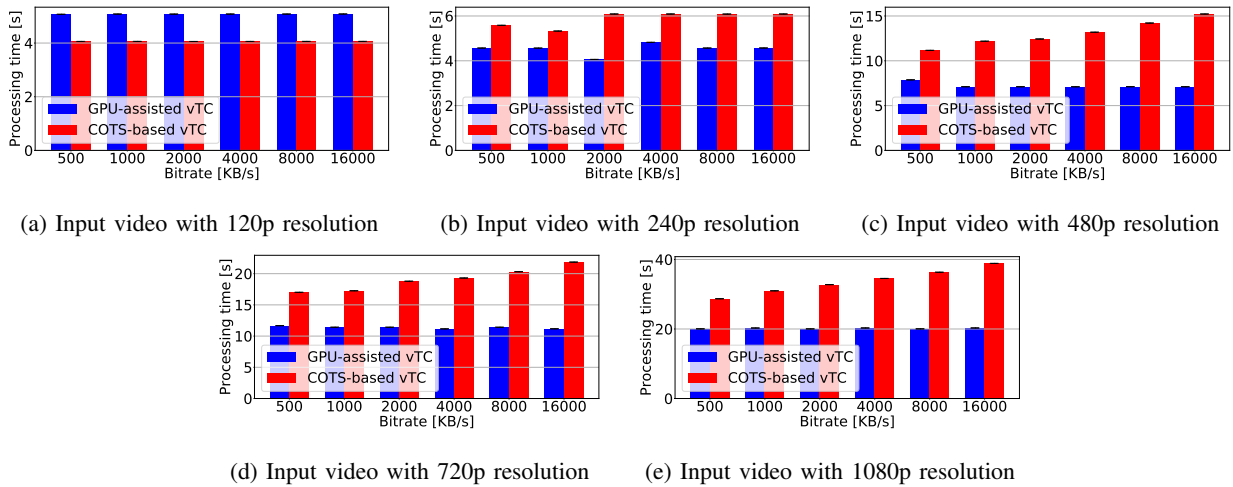


Fig. 1: Transcoding processing time for videos with different resolutions (with 95 % confidence interval - error bars are too small)

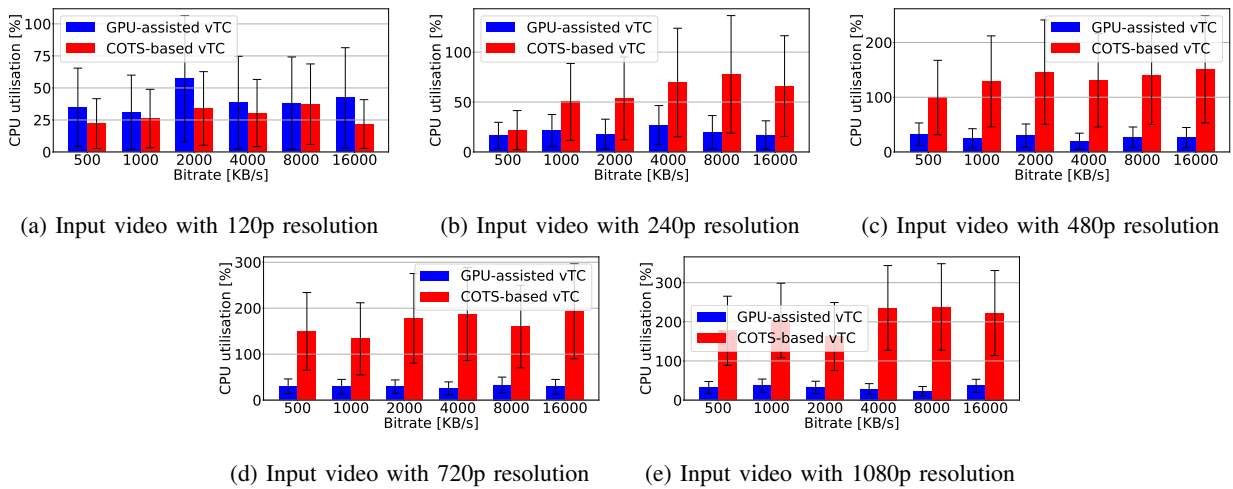


Fig. 2: Transcoding CPU utilization for videos with different resolutions (with 95 % confidence interval)

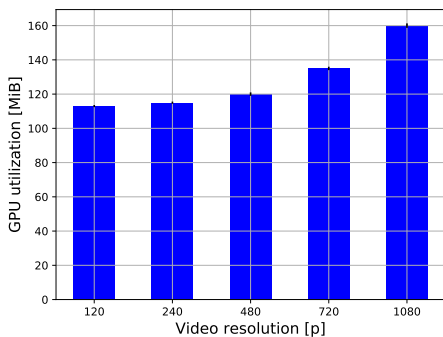


Fig. 3: Memory usage of GPU-assisted vTC for videos with different resolutions (with 95 % confidence interval - error bars are too small)

the processing time of vTC in some cases, in some other cases it is not a suitable implementation option as it increases the resource usage while not providing any performance improvement. This backs up the idea of multi-version services and dynamical service provisioning as using the right vTC version for the given input video can significantly improve performance and reduce resource usage.

B. Cost Analysis

We have also analysed the cost of running COTS-based and GPU-assisted vTCs. In the performance analysis, we observed how much CPU core and GPU memory are needed to run COTS-based and GPU-assisted vTCs, respectively. Having these data, we have looked at Amazon's EC2 price list³ to see how much it costs to provide these resources in a virtualized cloud environment. In our analysis, Amazon's general-purpose

³<https://aws.amazon.com/ec2/pricing/on-demand/>, accessed May 20, 2019

“t2” instances are considered as possible instances to provide required resources for vTCs. For the GPU case, we consider the price of elastic graphics instances “eg1”⁴ that allows GPU to be attached to EC2 instances.

The results of the cost analysis are illustrated in Fig. 3. It shows the cost of providing resources for different versions of the vTC for one hour. While the costs of GPU-assisted vTC remains constant for all video resolutions, the cost of COTS-based vTC shows an increasing trend as the video resolution increases. These results show that the use of COTS-based vTC for video with 1080p resolutions is inefficient both performance-wise and cost-wise. The same holds true for using GPU-assisted vTC to process videos with 120p and 240p resolutions. For the video with 480p and 720p resolutions, although GPU-based vTC performs better, it costs more compared to COTS-based vTC.

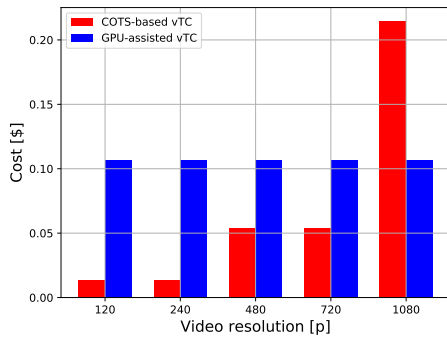


Fig. 4: Cost of running different versions of vTC on AWS resources for one hour

C. Management Overhead Analysis

Management overhead of multi-version services also needs to be considered. One of the main factors in multi-version service management is the service deployment time. In a static service deployment, deployment time does not play an important role as the service is deployed only once and the service usage is started once the service is deployed. But in the dynamic deployment scenario, we switch between different versions multiple times during the lifecycle of the service. In this situation, it is important to know how much delay version switching adds to the total latency of the service. To this end, we have also measured the deployment time of vTCs. We performed the measurement for two types of vTCs: (1) a Container-based vTC and (2) a VM-based vTC. A MANO framework called Pishahang [9] [10] has been utilized for this evaluation. Pishahang is an NFV multi-domain orchestrator that supports orchestration of VM- and Container-based services.

As shown in Fig.5, there is a significant deployment time difference between VM-based and Container-based vTCs.

While for CN-based vTC, it takes 2.57 seconds to get deployed, VM-based vTC takes more than one minute to get to the operational stage. Most of the deployment time for both container and VM is taken by the Virtual Infrastructure Manager (VIM) that spins up the VM or the container. The image downloading time is excluded from this time. From these results, we observe that in the case of using VM-based vTC, starting VM from scratch is not time-efficient and other solutions such as cold, warm, and live migrations [11] should be considered. We also observed that, from the deployment time point of view, CN-based vTCs are significantly better solutions for dynamic service deployment as they have negligible deployment time compared to VM-based vTC.

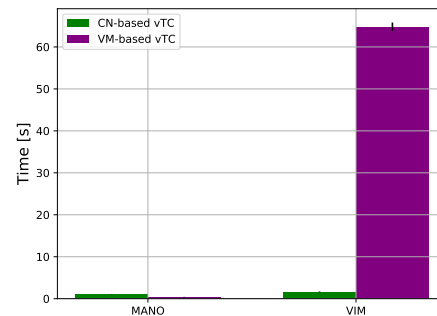


Fig. 5: The deployment time of different vTC versions (with 95 % confidence interval - error bars are too small)

IV. CONCLUSION

In this work, we observe that there are many VNFs that can benefit from dynamic deployment and that having such an approach can be used to improve the performance while guaranteeing the cheapest price for a particular service. Using vTC as a case study, we observe that using GPUs can accelerate the performance of vTC for high-quality videos while low-quality videos can be handled better by COTS-based vTCs and be cheaper at the same time. Our management overhead analysis shows that using VMs are not the best virtualization option when it comes to switching service versions on the fly as it generates a significant delay. On the other hand, containers seem to be a better virtualization environment as the deployment time of these resources is quite low.

ACKNOWLEDGMENT

This work has been partially supported by the 5G-PICTURE project, funded by the European Commission under Grant number 762057 through the Horizon 2020 and 5G-PPP programs and the German Research Foundation in the Collaborative Research Centre On-The-Fly Computing (SFB 901).

REFERENCES

- [1] L. Nobach, B. Rudolph, and D. Hausheer, “Benefits of conditional fpga provisioning for virtualized network functions,” in *2017 International Conference on Networked Systems (NetSys)*, March 2017, pp. 1–6.
- [2] N. Ghrada, M. F. Zhani, and Y. Elkhatib, “Price and performance of cloud-hosted virtual network functions: Analysis and future challenges,” in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, June 2018, pp. 482–487.

⁴<https://aws.amazon.com/ec2/elastic-graphics/>, accessed May 20, 2019

- [3] L. Nobach and D. Hausheer, "Open, elastic provisioning of hardware acceleration in nfv environments," in *2015 International Conference and Workshops on Networked Systems (NetSys)*, March 2015, pp. 1–5.
- [4] D. Thambawita, R. Ragel, and D. Elkaduwe, "To use or not to use: Graphics processing units (gpus) for pattern matching algorithms," in *7th International Conference on Information and Automation for Sustainability*. IEEE, 2014, pp. 1–4.
- [5] S. Dräxler and H. Karl, "SPRING: scaling, placement, and routing of heterogeneous services with flexible structures," in *2019 IEEE Conference on Network Softwarization (NetSoft)*, 2019.
- [6] S. Han, K. Jang, K. Park, and S. Moon, "Packetshader: a gpu-accelerated software router," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 195–206, 2011.
- [7] I. M. Araújo, C. Natalino, Á. L. Santana, and D. L. Cardoso, "Accelerating vnf-based deep packet inspection with the use of gpus," in *2018 20th International Conference on Transparent Optical Networks (ICTON)*. IEEE, 2018, pp. 1–4.
- [8] X. Li, M. A. Salehi, Y. Joshi, M. Darwich, B. Landreneau, and M. Bayoumi, "Performance analysis and modeling of video transcoding using heterogeneous cloud services," *CoRR*, vol. abs/1809.06529, 2018. [Online]. Available: <http://arxiv.org/abs/1809.06529>
- [9] H. R. Kouchaksaraei, T. Dierich, and H. Karl, "Pishahang: Joint orchestration of network function chains and distributed cloud applications," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, June 2018, pp. 344–346.
- [10] H. R. Kouchaksaraei and H. Karl, "Service function chaining across openstack and kubernetes domains," in *Proceedings of the 13th ACM International Conference on Distributed and Event-based Systems*, ser. DEBS '19. New York, NY, USA: ACM, 2019, pp. 240–243. [Online]. Available: <http://doi.acm.org/10.1145/3328905.3332505>
- [11] S. Thamarai Selvi, C. Valliyammai, G. P. Sindhu, and S. Sameer Basha, "Dynamic resource management in cloud," in *2014 Sixth International Conference on Advanced Computing (ICoAC)*, Dec 2014, pp. 287–291.