Price and Performance of Cloud-hosted Virtual Network Functions: Analysis and Future Challenges

Nadir Ghrada*, Mohamed Faten Zhani*, Yehia Elkhatib‡*
*École de Technologie Supérieure (ÉTS), Montreal, Quebec, Canada
‡MetaLab, School of Computing and Communications, Lancaster University, UK
E-mail: nadir.ghrada.1@ens.etsmtl.ca, mfzhani@etsmtl.ca, {i.lastname}@lancaster.ac.uk

Abstract*—The concept of Network Function Virtualization (NFV) has been introduced as a new paradigm in the recent few years. NFV offers a number of benefits including significantly increased maintainability and reduced deployment overhead. Several works have been done to optimize deployment (also called *embedding*) of virtual network functions (VNFs). However, no work to date has looked into optimizing the selection of cloud instances for a given VNF and its specific requirements. In this paper, we evaluate the performance of VNFs when embedded on different Amazon EC2 cloud instances. Specifically, we evaluate three VNFs (firewall, IDS, and NAT) in terms of arrival packet rate, resources utilization, and packet loss. Our results indicate that performance varies across instance types, departing from the intuition of "you get what you pay for" with cloud instances. We also find out that CPU is the critical resource for the tested VNFs, although their peak packet processing capacities differ considerably from each other. Finally, based on the obtained results, we identify key research challenges related to VNF instance selection and service chain provisioning.

I. Introduction

Since its introduction as a concept by ETSI to decouple software from hardware by leveraging virtualization technology [1], Network Function Virtualization (NFV) has been widely and rapidly adopted as more cost-effective and easy to manage replacement to traditional hardware-based middleboxes. Such monolithic hardware solutions are not just expensive to obtain and maintain, they also make it difficult to re-allocate network functions such as firewalls, load balancers, intrusion detection systems (IDS), and network address translation (NAT). In contrast, NFV enables such network functions to basically run on virtual machines (VMs) hosted by commodity servers as Virtual Network Functions (VNFs).

NFV offers numerous benefits to cloud service providers (CSPs) including networking equipment cost reduction, power consumption minimization, scalability, elasticity, hardware reuse, easy multi-tenancy, and rapid configuration of new services [1]. NFV services are composed of a set of network functions traversing the path(s) from one or multiple sources to the destination. Each such composition of ordered VNFs is referred to as a *service function chain* (SFC).

SFC embedding is an intensely active research area. However, little attention has been given to optimal selection of cloud resources – commonly the most popular infrastructure – for SFC embedding. This is a non-trivial challenge as several studies have identified that the 'book value' of cloud instances as reported by CSPs are unreliable as performance indicators [2], [3].

978-1-5386-4633-5/18/$31.00 ©2018 Crown

In this paper, we tackle the problem of identifying the optimal cloud instances for hosting VNF. We target Amazon EC2 as the market leading Infrastructure-as-a-Service (IaaS) provider. Through extensive experimentation, we analyze the performance-price trade-off of four different EC2 instance types. Our investigation distinguishes the problem into two main parts: instance selection and dimensioning. More specifically, the paper aims to answer the following questions:

1) Which instance type best meets the performance requirements of the SFC?
2) Which instance type provides the most cost-effective option for the constituent VNFs?
3) How many of instances are needed per VNF?

We are also mindful of the varying requirements of different VNFs. As such, we use 3 different VNFs as the use cases of our analysis, namely: firewall, IDS, and NAT.

The remainder of this paper is organized as follows. Section II reviews the state of the art. Section III presents the setup and results of our analysis, while Section IV discusses the outcomes and their significance. This is followed by Section V, which focuses a few key future challenges in light of our results, and Section VI concludes.

II. Related Work

There is a growing body of work on VNF placement and chaining (*e.g.,* [4], [5], [6]), and state management and migration (*e.g.,* [7], [8], [9], [10]). However, these approaches focus on theoretical optimization and overlook the performance of the physical infrastructure performance.

Reliance on the specifications of cloud instances as reported by CSPs is severely problematic as such specifications are not necessarily reliable. An early study [2] looked into variances between a few EC2 instance types using standard benchmarks and noted that there are no "best performing instance type". Other studies have identified variances within one provider [11], [12], [13], [14], and over a span of several days [15], different times of the year [16], and different regions of a single IaaS provider [17]. More recent efforts [3] indicate

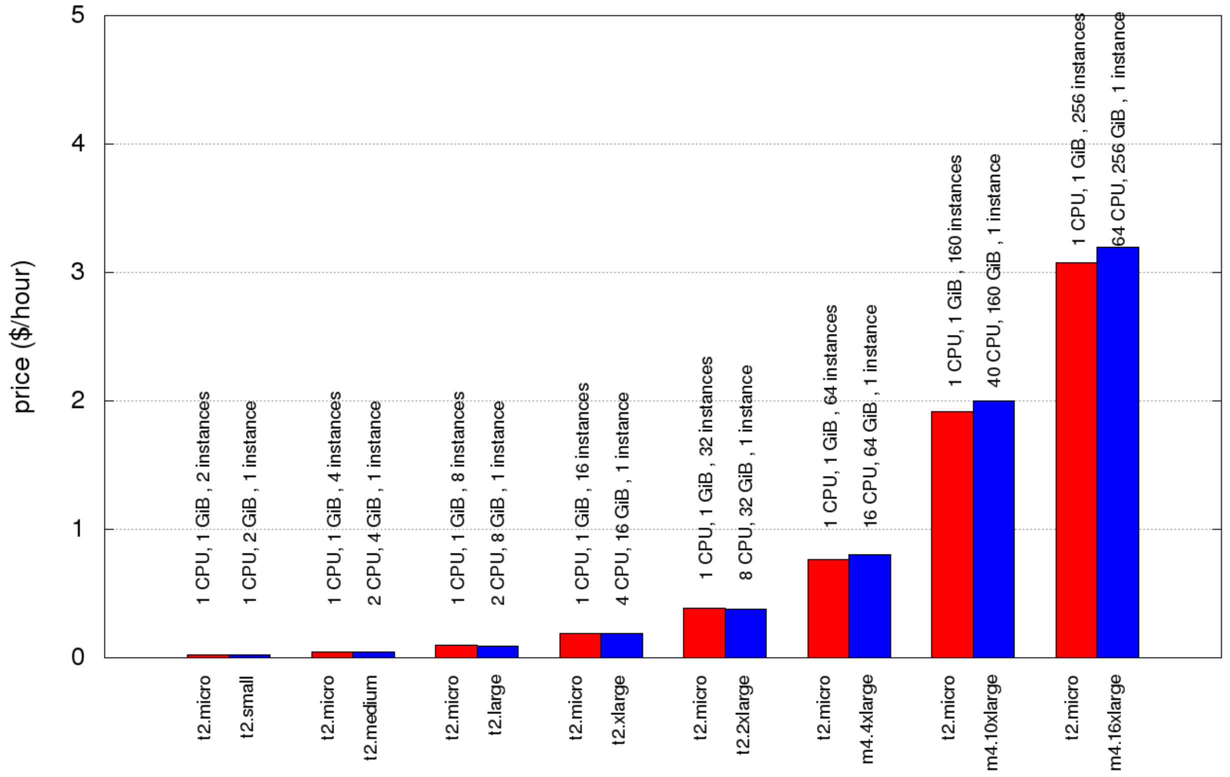


Fig. 1: Instance capacity versus Price

that cloud instance performance is difficult to foresee based on the information offered by CSPs (EC2 in particular). Thus, selecting an optimal instance is a non-trivial decision.

Therefore, some have tried to gain better understanding of the potential performance of cloud instances using standard or bespoke benchmarking suites and various modelling techniques; *e.g.,* [18], [19]. Others use profile-based methods (*cf.* [20], [21]) or application-specific performance models (*cf.* [22]). A recent paper looked into the suitability of containers for hosting VNFs [23]. However, no work to date has identified how to optimize the choice of cloud instances for VNFs, *i.e.,* ones that maximize service availability and packet processing rate, and minimize instance price.

III. Analysis of Instance Pricing and Performance

In this section, we analyze the prices of Amazon EC2 instances versus their size in terms of vCPU and memory. We then study the performance of different types of VNF instances using such instances in terms of a number of critical metrics: transmitted packets, packet arrival rate, CPU utilization and packet loss rate.

A. Price versus amount of resources

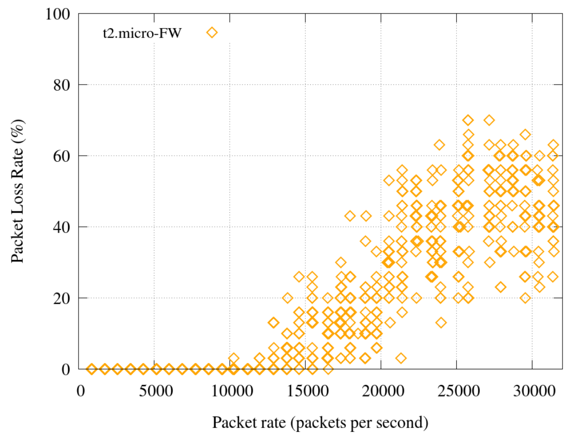
The cost of operating Amazon EC2 instances [24] is mainly reliant on their configuration (see Table I), *i.e.,* the amount of virtual resources (*i.e.,* vCPU and memory) allocated to run these instances.

TABLE I: Excerpt of the configuration and price of the EC2 instances, as of November 2017.

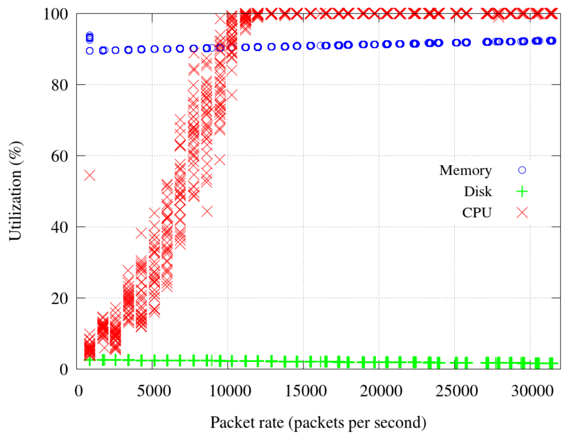
Instance Type	**vCPU Cores**	**Memory (GiB)**	**Price (US$/hour)**
`t2.micro`	1	1	0.012
`t2.small`	1	2	0.023
`t2.medium`	2	4	0.047
`t2.xlarge`	4	16	0.188

According to Amazon EC2 pricing list [24], the lowest instance configuration is `t2.micro`, which operates with the least resources in terms of CPU and memory (*i.e.,* 1 vCPU core and 1 GiB of memory). The price to run a single instance of `t2.micro`, for example, is $0.012 per hour whereas operating an instance of `t2.small` type, which is allocated a single vCPU and 2GiB RAM, is almost double the hourly rate at $0.023. This also means that having two `t2.micro` instances would cost a tenant $0.024 per hour. However, the tenant will have twice the processing capacity of a single `t2.small`. In other words, two `t2.micro` instances would *in theory* provide better performance than a single `t2.small` instance with merely the same price.

To better analyze whether this holds true for other types of EC2 instances, we evaluate how many `t2.micro` instances it would take to provide the same amount of resources of larger EC2 instance types, and how do they compare in terms of price.

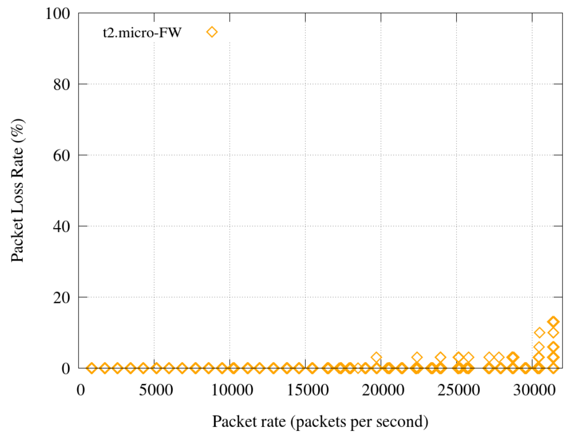


(a) Packet loss

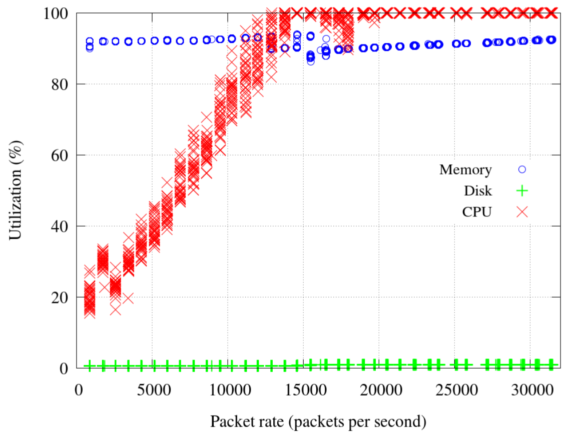


(b) Resource utilization

Fig. 2: Firewall

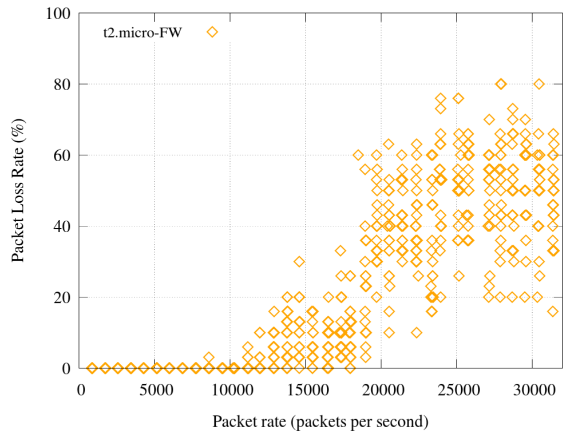


(a) Packet loss

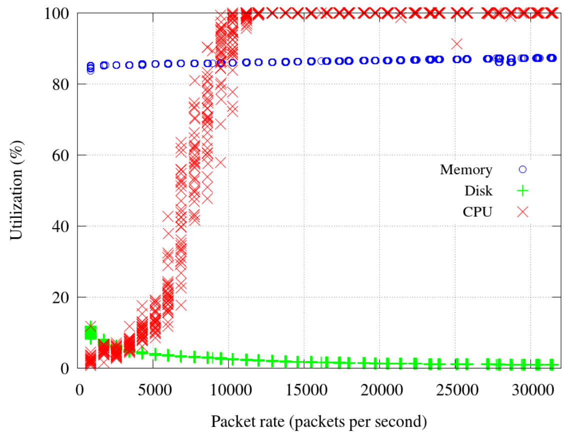


(b) Resource utilization

Fig. 3: IDS



(a) Packet loss



(b) Resource utilization

Fig. 4: NAT

Fig. 1 shows the cost of each EC2 instance as well as the number and price of the `t2.micro` instances that would provide an equivalent amount of resources in terms of CPU and memory. For example, the largest EC2 instance, `m4.16xlarge`, provides 64 vCPU cores and 256 GiB, and costs \$3.2/hour. The same amount of memory could be provided by 256 `t2.micro` instances with a cost of \$3.1/hour, but with significantly more vCPU cores (256 compared to 64). The same observation holds for all other types of instances suggesting that using smaller instances (*i.e.,* `t2.mirco`) would, theoretically, provide reduced costs and increased performance especially in terms of CPU horsepower.

B. VNF performance versus instance type

In this subsection, we evaluate the performance of three VNFs namely a firewall, an IDS, and a NAT. We ran Shorewall as a firewall [25], Snort as a network IDS [26], and the Ubuntu-based NAT.

• **Experimental Setup:** We start the evaluation by generating UDP traffic between a pair of instances, where the VNF is installed inline to face the incoming traffic, process it, and then forward it to the destination. We ran the same experiment to examine the processing capacity of all VNFs.

We generated traffic using `iperf`, with this traffic increasing linearly with time. The purpose of increasing traffic load is to measure the CPU, memory, and disk utilization as well as packet loss for a VNF instance for different packet arrival rates. By doing this, we determine the highest packet processing capacity that each VNF can process per second in a particular instance.

• **Micro instance processing capacity:** As the previous subsection highlighted, the `t2.micro` instance offers the most cost-effective access to VNF-hosting cloud instances. Thus, we look at how capable such instance is at hosting VNFs.

For each VNF, we measure CPU, memory and disk utilization as well as packet loss as the received traffic increases over time. The results are depicted in Fig. 2–4. For the firewall VNF running on a `t2.micro` instance, it becomes overloaded and rejects packets from around 11,000 packets per second (Fig. 2(a)). This corresponds to 90% CPU and memory utilization (Fig. 2(b)).

We also see in Fig. 3 that an IDS running on a `t2.micro` instance behaves differently from a firewall VNF on the same instance. Packet losses start to occur when more than 18,000 packets start to arrive per second (Fig. 3(a)). This corresponds to 100% and 90% of CPU and memory utilization (Fig. 3(b)), respectively.

Finally, the results for the NAT are reported in Fig. 4. We note that the performance of the `t2.micro`-hosted NAT starts to deteriorate around 9,000 packets per second arrival rate (Fig. 4(a)). This corresponds to 90% CPU and memory utilization (Fig. 4(b)).

• **Amount of resources versus performance:** To compare the performance of instances of different sizes, we ran the same experiments for different instance types. For each of them, we identify the peak packet processing capacity (packets per second) as the maximum packet processing rate of the VNF when the CPU utilization reaches 90% or the packet loss reaches 10%, whichever first.

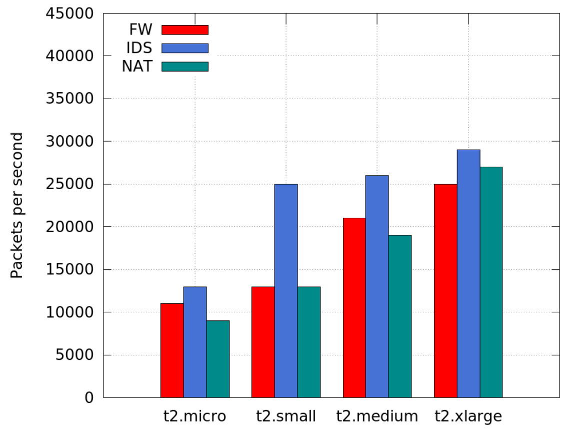


Fig. 5: VNF peak packet service rate per instance type, as dictated by CPU utilization of 90% or packet loss of 10%, whichever first

Fig. 5 summarizes the results and compares the packet processing capacity of four types of instances, `t2.micro`, `t2.small`, `t2.medium` and `t2.xlarge` for the three VNFs. The first observation is that the packet processing capacity may significantly vary depending on the type of the network function deployed in the instance. For example, the figure shows that the `t2.micro` can process as high as 13,000 packets/second when used to host a firewall, 13,000 packets/second for the IDS, and as low as 9,000 packets/second for the NAT.

We also notice that increasing the size of the instance results in a higher processing capacity. However, surprisingly, the increase in processing capacity is not proportional to the number of CPU cores, nor to its price. For instance, int he case of a NAT (Fig. 5), a `t2.micro` instance (1 CPU, 1 GiB, \$0.012/hour) can process 9,000 packets/second whereas a `t2.xlarge` (4 CPU, 16 GiB, \$0.188/hour) can process 26,000 packets/second; only an 188% increase for more than 15x the price. In other words, even though `t2.xlarge` has 4 times more CPUs and 16 times more memory than `t2.micro`, it can process only 2.8 times more packets than a `t2.micro` instance. This suggests that deploying several `t2.micro` instance provides better processing performance. Finally, recall that the price of 16 `t2.micro` instances is less than a single `t2.xlarge` (Fig. 1), we can hence conclude that `t2.micro` instances are also more appealing in terms of cost-effectiveness

IV. Discussion

Abstracting from the above observations, we extract the following outcomes.

A. Input packet rate is crucial; CPU is the bottleneck

The incoming packet rate is the determining factor in the performance of VNFs. Consequently, anticipating such packet rate and its variance is a key element of dimensioning VNF embedding across any network.

Based on the results plotted in Fig. 2–4, VNF packet processing rates could deteriorate due to increased contention on the virtual CPU resource. This is the case for all tested VNF types. Of course, an interesting avenue of future work would be to ascertain if this holds for a wider range of VNFs.

B. Not all VNFs are the same

Every deployment of a VNF has its own performance 'breaking point', which varies across VNF types and we conjecture that it would vary also between different implementations of the same VNF. We observed that a low-demand VNF such as an IDS is easily hosted on a very modest cloud instance such as the EC2 `t2.micro`. However, to obtain similar packet rate capacity for a slightly more demanding VNF, such as a firewall, it requires hosting on a higher specification instance (in this case, `t2.medium`).

We can hence conclude that the type of the VNF and its implementation (in other words, the software running on the instance) will have a major impact on the packet processing capacity of the VNF. It is therefore a major challenge to write customized code that can optimize the VNF operations and further leverage the resources available in the instance where the network function is running.

C. Instance specification is a loose indicator of performance

As described in Section II, this observation has already been made by different research groups using both generic benchmarks and specific application profiles. This has been a motivator for our study, and indeed we have verified that such variance does exist even for NFV workloads. We found also the performance is not proportional to the amount of CPU and memory allocated for the VNF. This makes selecting the right instance based on the amount of the instance resources not accurate enough. As a result, the instance specification is only a loose indicator of its performance. It is therefore of utmost importance to test in advance the VNF when running on a particular instance, and evaluate its packet processing capacity as it may vary depending on the instance type.

To conclude, we can say that current cloud provider offering models that provide generic VM instances with predefined resource capacity (*i.e.,* CPU, memory and disk) are not perfectly suitable for VNFs. A better offering model would be to provide VNFs with a well-defined network function software (software name and version) and an average packet processing capacity (in terms of packets per second).

D. Micro instances are more cost-effective

The obtained results clearly show that, compared to large instances, deploying multiple micro instances provides not only much more resources (in terms of CPU, memory and disk), but also much more packet processing capacity with much less price. By combining results of Fig. 1 and Fig. 5, we can compare the performance, amount of resources and prices for a single `t2.xlarge` instance and 16 `t2.micro` instances. As shown in Table II, for the same price, we can provision 16 `t2.micro` instances that provides 6 times the packet processing capacity of a single `t2.xlarge`.

TABLE II: Firewall performance using a single `t2.xlarge` instance compared to 16 `t2.micro` instances.

Instance Type	**vCPU Cores**	**Memory (GiB)**	**Price (US$/hour)**	**Packet Processing Capacity (pckts/s)**
`t2.xlarge`	4	16	0.1856	26,500
`t2.micro`	16	16	0.1856	144,000

V. Future Research Challenges

Based on these outcomes, we foresee future challenges in this domain to center around the following.

A. VNF instance selection and placement

Instance selection process is a daunting challenge that demands good knowledge of the VNF operational requirements and the performance of the hosting infrastructure [27]. The former requires analyzing the function operations, but the latter calls for a data-driven understanding of the performance of the underlying hosting infrastructure, and how this might vary over time. Indeed, hosting infrastructures are composed of several networking equipment, links and servers that are highly heterogeneous in capacity and performance. As a result, the performance of an instance with a predefined resource capacity might significantly vary in practice from one hosting server to another and from one network to another. Deciding of the instance type and its placement should therefore take into account the dynamic performance of the hosting equipment over time and space. The decision also depends heavily on the anticipated packet rate to be serviced. Therefore, a central challenge related to cloud-based VNF deployment is to put forward data-driven instance selection and placement solutions that are dynamic and adaptive.

A caveat here is that such data-driven approaches require significant amount of data to learn about the hosting infrastructure performance levels and their variations. Such data, obviously, comes at a cost, especially for large-scale infrastructures. Consequently, further efforts are needed to reduce such costs through either improved machine learning techniques (*e.g.,* [28]) or through data augmentation from external sources such as CloudHarmony [29].

B. Dependency-aware service chain embedding

In this work, we have only considered the performance-price tradeoff for hosting a single VNF in the cloud. This tradeoff naturally becomes significantly more complex for a SFC with more than one VNF as the decision for any single VNF would be affected by that of other VNFs in the same chain. In this case, it is challenging to satisfy the requirements of all VNFs in the chain while ensuring their dependency requirements (*e.g.,* packet rate, delay).

Furthermore, our results suggest that micro-instances are more cost-effective than large ones. However, deploying the same function over several micro-instances is a non-trivial challenge as, depending on the type of the function and its requirements, some data might need to be shared among these instances. In this case, efficient synchronization techniques between the VNF instances need to be devised and evaluated to ensure normal operation of the overall network function. A typical example is that of an IDS. On one hand, when it is provisioned over a single large instance, there is no need for synchronization; however, the instance is expensive. On the other hand, when several micro-instances of an IDS are deployed, they are able to process much more traffic (6x according to Table II) at a cheaper price. The instances analyze different shares of the incoming traffic so they need to pool information about detected attacks and incident analysis in order to make sure potential threats are detected.

C. Designing Serverless-based NFV-based service chains

At a conceptual level, NFV lends itself to deployment using a serverless or Function-as-a-Service (FaaS) fashion. This is because many VNFs are relatively small, self-contained functions that are easily deployable across different locations in the network. More importantly, though, their operational cost depends heavily on the incoming service rate, as already highlighted. As such, they stand to benefit from FaaS deployment, which enables greater elasticity and flexibility in the face of highly variable packet rates [30]. In this context, a major challenge would be to design serverless-based NFV service chains that are dynamically provisioned and torn down depending on the demand, the traffic routes, and the available resources in the underlying infrastructures.

VI. Conclusion

In this paper, we addressed the question of identifying the most suitable cloud instances for hosting VNFs, and the effect of such decision on the VNF price and performance. We studied three different popular middlebox functions: firewall, IDS, and NAT. Our experimentation spanned four different instance types from Amazon EC2. From our experiments, we surmise that naïve hosting based on instance specification alone is suboptimal and costly. Multiple smaller instances provide much better performance-to-price ratio than a single large instance. Based on our results, we discussed key research challenges to provide automated VNF instantiation and deployment solutions, and to investigate how this could affect service chain provisioning in cloud-based infrastructures.

Acknowledgment

This work was partly supported by the Research Internationalization Fund at École de Technologie Supérieure (ÉTS) Montreal; and by the Adaptive Brokerage for the Cloud (ABC) project, UK EPSRC grant EP/R010889/1.

References

[1] M. Chiosi *et al.*, "Network functions virtualisation," *SDN and OpenFlow World Congress*, 2012.

[2] S. Ostermann *et al.*, "A performance analysis of EC2 cloud computing services for scientific computing," in *Conf. on Cloud Computing (CloudComp)*, 2009.

[3] F. Samreen, Y. Elkhatib, M. Rowe, and G. S. Blair, "Daleel: Simplifying cloud instance selection using machine learning," in *Network Operations and Management Symposium (NOMS)*, 2016.

[4] M. Ghaznavi *et al.*, "Elastic virtual network function placement," in *Conf. on Cloud Networking (CloudNet)*, 2015.

[5] W. Racheg, N. Ghrada, and M. F. Zhani, "Profit-driven resource provisioning in NFV-based environments," in *IEEE Conf. on Communications (ICC)*, 2017.

[6] F. Carpio, S. Dhahri, and A. Jukan, "VNF placement with replication for load balancing in NFV networks," in *IEEE Conf. on Communications (ICC)*, 2017.

[7] B. Kothandaraman, M. Du, and P. Sköldström, "Centrally controlled distributed VNF state management," in *HotMiddlebox*, 2015.

[8] M. Peuster and H. Karl, "E-state: Distributed state management in elastic network function deployments," in *IEEE Conf. on Network Softwarization (NetSoft)*, 2016.

[9] M. F. Zhani, Q. Zhang, G. Simon, and R. Boutaba, "VDC Planner: Dynamic migration-aware virtual data center embedding for clouds," in *IEEE/IFIP Integrated Network Management Symposium (IM)*, 2013.

[10] R. Boutaba, Q. Zhang, and M. F. Zhani, "Virtual machine migration: Benefits, challenges and approaches," *Communication Infrastructures for Cloud Computing: Design and Applications*, pp. 383–408, 2013.

[11] K. R. Jackson *et al.*, "Performance analysis of high performance computing applications on the amazon web services cloud," in *Conf. on Cloud Computing Technology and Science (CloudCom)*, 2010.

[12] Z. Ou *et al.*, "Exploiting hardware heterogeneity within the same instance type of amazon EC2," in *HotCloud*, 2012.

[13] Z. Li, L. O'Brien, R. Ranjan, and M. Zhang, "Early observations on performance of Google compute engine for scientific computing," in *Conf. on Cloud Computing Technology and Science (CloudCom)*, 2013.

[14] K. Hwang *et al.*, "Cloud performance modeling with benchmark evaluation of elastic scaling strategies," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, pp. 130–143, 2016.

[15] Z. Li, L. OBrien, and H. Zhang, "CEEM: A practical methodology for cloud services evaluation," in *World Congress on Services*, 2013.

[16] J. Schad, J. Dittrich, and J.-A. Quiané-Ruiz, "Runtime measurements in the cloud: Observing, analyzing, and reducing variance," *Proc. VLDB Endow.*, vol. 3, no. 1-2, 2010.

[17] J. L. Lucas-Simarro *et al.*, *A Cloud Broker Architecture for Multicloud Environments*, 2013, pp. 359–376.

[18] S. C. Phillips, V. Engen, and J. Papay, "Snow white clouds and the seven dwarfs," in *Conf. on Cloud Computing Technology and Science (CloudCom)*, 2011.

[19] J. OLoughlin and L. Gillam, "Towards performance prediction for public infrastructure clouds: An EC2 case study," in *Conf. on Cloud Computing Technology and Science (CloudCom)*, 2013.

[20] A. Li, X. Yang, S. Kandula, and M. Zhang, "CloudCmp: Comparing public cloud providers," in *SIGCOMM Conf. on Internet Measurement (IMC)*, 2010.

[21] B. Varghese, O. Akgun, I. Miguel, L. Thai, and A. Barker, "Cloud benchmarking for performance," in *Conf. on Cloud Computing Technology and Science (CloudCom)*, 2014.

[22] S. Venkataraman, Z. Yang, M. Franklin, B. Recht, and I. Stoica, "Ernest: Efficient performance prediction for large-scale advanced analytics," in *Symp. on Networked Systems Design and Implementation (NSDI)*, 2016.

[23] J. Struye, B. Spinnewyn, K. Spaey, K. Bonjean, and S. Latre, "Assessing the value of containers for NFVs: A detailed network performance study," in *Conf. on Network and Service Management (CNSM)*, 2017.

[24] "Amazon EC2 pricing on demand," https://aws.amazon.com/ec2/pricing/on-demand/, [accessed 20-Jan-2017].

[25] "Shorewall Firewall," http://shorewall.org, [accessed 22-Jun-2017].

[26] "Snort," https://www.snort.org/, [accessed 19-Feb-2017].

[27] Y. Elkhatib, "Mapping Cross-Cloud Systems: Challenges and Opportunities," in *HotCloud*, 2016.

[28] F. Samreen, G. S. Blair, and Y. Elkhatib, "Transferable knowledge for low-cost decision making in cloud environments," 2018.

[29] "CloudHarmony," https://cloudharmony.com/, [accessed 23-Jan-2018].

[30] I. Baldini *et al.*, "Serverless computing: Current trends and open problems," *CoRR*, vol. abs/1706.03178, 2017.