An Optimization Case in Support of Next Generation NFV Deployment

Zahra Abbasi, Ming Xia M, Meral Shirazipour and Attila Takacs
Ericsson Research Silicon Valley

1 Introduction and motivation

Still not long ago operators were struggling with middlebox deployment and traffic management across them. The service chaining problem was a well studied subject which had to deal with the limitations of middleboxes and offer various techniques to overcome them to achieve the desired traffic steering [6, 7]. Only two years have passed its official launch by ETSI [1], but network function virtualization (NFV) has already revolutionized the telecom industry by proposing a complete design paradigm shift in the way middleboxes are built and deployed. NFV requires the virtualization of middleboxes and other networking equipment, called virtual network functions (vNFs). This requirement will allow networking infrastructure operators to benefit from the same economies of scale and flexibility than the information technology community experienced with the advent of cloud computing. Other than the capex/opex saving and faster time to market of new services, the cloudification of networking gives us the opportunity to rethink how networking equipment are designed and deployed.

While first generation NFV deployment has already been proved in the ETSI NFV community, we argue that there is yet lots of space for improvement both in terms of vNF design and deployment in order to optimize cloud computing and networking resources. To this end, we foresee various redesign opportunities for vNFs, either in distributing the functional components of a single vNF across the cloud to achieve better performance and scale, or in running multiple instances of vNFs in smaller VMs, as proposed by ClickOS-NFV [5]. Such vNF designs open new horizons to perform optimized vNF deployment in the cloud as studied in this paper.

This position paper takes the example of ClickOS-NFV [5] and proposes a clever way to address the vNF placement in the cloud and vNF forwarding graph (vNF-FG) problem, a name given by ETSI to the well-known *service chaining* problem which is the process of steering a traffic flow across a predefined set of middleboxes (now vNFs). Our proposed vNF chaining and placement optimization model and algorithm shows the benefits of such next generation NFV deployment. Moreover, our model, managing both cloud computation and networking resources, has more room for optimization as opposed to existing cloud based NFV deployment solutions, such as Stratos [4], which offer to auto-scale vNFs merely depending upon the input workload.

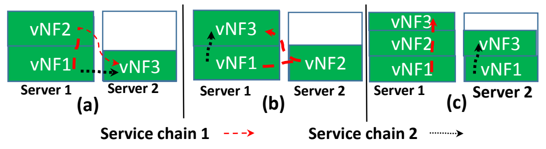

Figure 1: Example showing tradeoff of computation and communication overhead

Computation vs communication overhead: Depending on the vNF chaining policy, granular per user/application flows will go through a sequence of vNFs. The communication overhead (e.g., link bandwidth usage, traversal delay, and switches' TCAM resources [3, 6]) of a vNF chain often directly depends on the number of hops in the path. Ideally operators would like to minimize the number of vNF instances running in their cloud in order to reduce the computation overhead (e.g., VM/container overhead and software cost which is typically given per a running instance). However, consolidating a vNFs' load into small number of instances may increase the communication overhead.

For example, consider three vNFs: vNF_1, vNF_2, and vNF_3, and two ordered vNF chains: (1) $< vNF_1, vNF_2, vNF_3 >$, and (2) $< vNF_1, vNF_3 >$. Also assume all vNFs have equal computation load and that every two of them with full load (load from both of the service chains) can fit into a single physical server. Figure 1 shows three placement solutions of these vNFs. Placement (a) causes both vNF chains to traverse one hop. Placement (b) causes vNF chain 2 to visit only one node (no hop) but the vNF chain 1 needs to traverse two hops which worse compared to placement (a). On the contrary, in placement (c) both vNF chains only need to visit one node (no hop) which is an ideal case from the communication perspective. However the computation cost in (c) is more than those of (a) and (b).

2 Problem formulation and algorithm

The problem of joint optimization of vNF placement and workload (traffic) distribution is formulated in this section. Note that such flexibility is only possible with next generation vNFs such as [5]. Given a cloud with N number of nodes to implement M vNF chains (with possible vNF orders $S_{m,k}$, since a vNF chain m can be partially or fully order-insensitive), we would like to place vNFs in the cloud to implement the given vNF chains and load balance the fraction of workload per chain that visits each vNF such that the sum of the computation and the communication overhead is minimized. The problem is formulated as:

Sets:

· I, the set of vNF types (indexed by i);

· M, the set of vNF chains to be placed (indexed by m);

· N, the set of nodes (servers) available to the operator (indexed by n);

· $S_{m,k}$, the set of possible vNF permutations per requested M, $S_{m,k} = s_1 \ldots s_j \ldots$, $s_j \in I$ and j is the position of the vNF s_j in kth permutation of vNF chain m;

Constants:

· Λ_m, the workload of vNF chain $m \in M$;

· C_n, the capacity of node $n \in N$;

· u_i, the extra computation cost that vNF type i imposes per unit input workload, $i \in I$;

· α_i, the computation cost per vNF instance of type i;

· β, the communication cost per hop that a vNF chain crosses;

Variables:

· $x_{i,n}$, a binary variable such that vNF type i resides in node n;

· $\lambda_{i,m,n}$, a variable indicating the fraction of workload from vNF chain m assigned to vNF type i in node n;

· $h_{s,k,m,n}$, a binary variable indicating whether all workload to vNF type s hosted in node n coming from vNF chain (k,m) is local;

Objective function:

$$\text{Minimize:} \quad \beta \min_k \sum_n \sum_{(s,s') \in S_{m,k}} h_{s,k,m,n} + \sum_n \sum_i \alpha_i x_{i,n}, \tag{1}$$

Constraints:

$$\begin{aligned} &\textstyle\sum_n \lambda_{i,m,n} = \Lambda_m, && \forall m,i \\ &\textstyle\sum_m \sum_i \lambda_{i,m,n} u_i \le C_n, && \forall n \\ &x_{i,n} \in \{0,1\}, x_{i,n} \ge \frac{\lambda_{i,n}}{\sum_m \Lambda_m}, && \forall i,n \end{aligned} \tag{2}$$

$$\begin{aligned} &h_{s,k,m,n} \ge \lambda_{s,k,n} - \lambda_{s',k,n}, \\ &h_{s,m,k,n} \in \{0,1\}, \forall n \in N, \text{and every two subsequent vNFs} \\ &(s',s) \in S_{m,k}. \end{aligned} \tag{3}$$

Computation Cost: Equations of (2) calculate $x_{i,n}$, from which the number of required instances per each vNF and consequently the computation cost is calculated.

Communication cost: We consider that for every two subsequent $(s',s) \in S_{m,k}$, if all the input workload of vNF s comes from the local node, i.e., s' is also hosted in the same node n, and the input workload to s is less than or equal to that of s', then the flows of chain m for that specific vNF instance do not require any external hop visit. For a given sequence k of a vNF chain m, denote by binary variable $h_{s,k,m,n}$ to indicate whether input flows to vNF type of s hosted in node n requires to visit a hop in order to get served. We can calculate the value of binary variable h as in equation (3). Then the total number of hops that the service chain m visits is equal to: $\min_k \sum_n \sum_{(s,s') \in S_{k,m}} h_{s,m,k,n}$. That is, the solution considers the vNF chain permutation where the required number of hops is minimal. The communication cost can be calculated form the total number of hops that all vNF chains visit.

Algorithm and preliminary evaluation: The Mixed Integer Linear Programming (MILP) formulation above introduces integer knobs to keep track of number of instances per vNF (i.e., x) and the number of hops a service chain visits (i.e., h), in order to account for computation and communication cost, respectively. In this paper we solve the problem (1) using branch and bound (MILP), and a heuristic greedy solution (Greedy). The basic idea of Greedy solution is to solve the vNF deployment problem for every vNF chain one by one given the previous vNF placement decisions. The order in which vNF chains are solved have big impact on the performance of the solution against MILP. For that we first order vNF chains such that those who has more common number of vNFs are solved next to each other. For some preliminary results we use GLPK [2] to implement and compare the MILP and Greedy solutions' scalability. We consider 3 vNF chains each at most consisting of 5 vNF. We run the algorithms with 5, 10, and 20 nodes (i.e., N). Greedy solution in all cases runs in fraction of a second. MILP runs in fraction of a second for $N = 5$, a less that 10 seconds for $N = 10$, and around half an hour for $N = 20$. The faster speed of Greedy, however, comes with increasing the communication cost. We ran the experiments on a 1.8 GHz Intel core i5, and 8 GB memory. The paper positions optimization as a case for next generation vNF deployment. As future works we need to design a near optimal but scalable algorithm for use in cloud-scale environment, and also compare the cost results between first and newer generation vNFs.

References

[1] ETSI. Leading operators create ETSI standards group for network functions virtualization. http://www.etsi.org/index.php/news-events/news/644-2013-01-isg-nfv-created. Accessed: 2015-03-17.

[2] GLPK - GNU Project - Free Software Foundation (FSF). https://www.gnu.org/software/glpk/. Accessed: 2015-01-01.

[3] FAYAZBAKHSH, S. K., CHIANG, L., SEKAR, V., YU, M., AND MOGUL, J. C. Enforcing network-wide policies in the presence of dynamic middlebox actions using flowtags. In *Proc. NSDI* (2014).

[4] GEMBER, A., GRANDL, R., ANAND, A., BENSON, T., AND AKELLA, A. Stratos: Virtual middleboxes as first-class entities. *UW-Madison TR1771* (2012).

[5] MARTINS, J., AHMED, M., RAICIU, C., OLTEANU, V., HONDA, M., BIFULCO, R., AND HUICI, F. Clickos and the art of network function virtualization. In *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14). Seattle, WA: USENIX Association* (2014), pp. 459–473.

[6] QAZI, Z. A., TU, C.-C., CHIANG, L., MIAO, R., SEKAR, V., AND YU, M. Simple-fying middlebox policy enforcement using sdn. In *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM* (2013), ACM, pp. 27–38.

[7] ZHANG, Y., BEHESHTI, N., BELIVEAU, L., LEFEBVRE, G., MANGHIRMALANI, R., MISHRA, R., PATNEY, R., SHIRAZIPOUR, M., SUBRAHMANIAM, R., TRUCHAN, C., ET AL. Steering: A software-defined networking for inline service chaining. In *ICNP* (2013), pp. 1–10.