BPA: The Optimal Placement of Interdependent VNFs in Many-Core System

Youbing Zhong[1,2,3], Zhou Zhou[1,2,3(✉)], Xuan Liu[4], Da Li[5], Meijun Guo[6], Shuai Zhang[1,2,3], Qingyun Liu[1,2,3], and Li Guo[1,2,3]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
zhouzhou@iie.ac.cn
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3] National Engineering Laboratory of Information Security Technologies, Beijing, China
[4] School of Information Engineering, Yangzhou University, Yangzhou, China
[5] Department of Electrical and Computer Engineering, University of Missouri, Columbia, USA
[6] School of Mathematical Sciences, University of Chinese Academy of Sciences, Beijing, China

Abstract. Network function virtualization (NFV) brings the potential to provide the flexible implementation of network functions and reduce overall hardware cost by running service function chains (SFCs) on commercial off-the-shelf servers with many-core processors. Towards this direction, both academia and industry have spent vast amounts of effort to address the optimal placement challenges of NFV middleboxes. Most of the servers usually are equipped with Intel X86 processors, which adopt Non-Uniform Memory Access (NUMA) architecture. However, existing solutions for placing SFCs in one server either ignore the impact of hardware architecture or overlook the dependency between middleboxes. Our empirical analysis shows that the placement of virtual network functions (VNFs) with interdependency in a server needs more particular consideration. In this paper, we first manage the optimal placement of VNFs by jointly considering the discrepancy of cores in different NUMA nodes and interdependency between network functions (NFs), and formulate the optimization problem as a Non-Linear Integer Programming (NLIP) model. Then we find a reasonable metric to describe the dependency relation formally. Finally, we propose a heuristic-based backtracking placement algorithm (BPA) to find the near-optimal placement solution. The evaluation shows that, compared with two state-of-art placement strategies, our algorithm can improve the aggregate performance by an average of 20% or 45% within an acceptable time range.

Keywords: NFV · SFCs plaecment · Interdependent NFs · NUMA

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2021
Published by Springer Nature Switzerland AG 2021. All Rights Reserved
H. Gao et al. (Eds.): CollaborateCom 2020, LNICST 350, pp. 305–319, 2021.
https://doi.org/10.1007/978-3-030-67540-0_18

1 Introduction

The emerging of Network Function Virtualization (NFV) has been an innovative technology to network architecture in recent years [1]. It is devoted to addressing the limitations of traditional middleboxes [2]. Unlike the tradition network architecture hosted on dedicated physical equipments or customized hardware, NFV runs network functions (NFs) using the software on the commodity servers with general processors, such as Intel X86, by leveraging underlying virtualization technology. What's more, NFV brings substantial flexibility for the distribution of network service and cost reduction. Telecom operators can deploy NFs according to requirements dynamically. A network service is usually composed of a series of NFs, which are often referred to as Service Function Chain (SFC). The traffic generated by request traverses the NFs one by one.

To achieve desired performance for NFV, the commodity servers to host Virtualized Network Functions (VNFs) are equipped with high-performance processors with many cores, which we refer to as many-core systems [3]. This kind of powerful server usually has enough capacity to accommodate an entire SFC or even multiple SFCs completely [4].

In the current stage, most of the studies focus on the placement of SFCs across multiple physical servers [5,6]. Almost all of the researches treat each server as an entire node. That's to say, all cores in the server are equal in the allocation of various NFs. However, CPU with many cores usually adopts Non-Uniform Memory Access (NUMA) architecture. For instance, most of Intel x86 CPU cores are partitioned into several nodes. Each node contains multiple cores and its own local memory. The performance of assigning CPU cores to host SFC under cross-node and Intra-node cases is significantly different. One of the main factors is that local memory access is much faster than remote access through Intel QuickPath Interconnect (QPI).

Furthermore, the placement of SFCs is also constrained by the dependency relation that may or may not exist between NFs [2]. For instance, there may or may not exist dependency between middleboxs. For instance, the IPSec decryptor is usually deployed before a NAT gateway [7], while the VPN proxycan be deployed either before or after a firewall [8]. Besides, NFs are usually heterogeneous and diverse. Different NFs have different computation resource requirements. The dependency relation of different types of middleboxes further complicates the placement of SFCs. We refer the NFs with interdependency as NF couple (two NFs).

We use the following example to illustrate the dependency relation effects for SFCs placement problem. In this paper, we treat all CPU cores in one NUMA node as equal. Consider a SFC including six NFs co-locating on two nodes as shown in Fig. 1(a). Here, we first consider this chain with computation cost: NF1 (100 units), NF2 (300 units), NF3 (300 units), NF4 (200 units), NF5 (100 units), NF6 (200 units). In this paper, we assume that the flow path is predetermined(NF1 $\rightarrow$ NF2 $\rightarrow$ NF3 $\rightarrow$ NF4 $\rightarrow$ NF5 $\rightarrow$ NF6). Namely, there exists dependency between each other, and all of NFs are totally-ordered. Each node has 600 computation units. For separated NFs, this placement makes use of the

most available resources on the nodes. NF1, NF2, and NF4 share LLC and local memory in the local node and NF3, NF5, NF6 can be considered as sharing QPI with the remote node (Fig. 1(b)). However, if there exists a sequential order between NF2 and NF3, namely flow traversing through NF2 before NF3, cross-node chaining between these two NFs can change the directly available resource subsequent NFs can get. We can see that there almost no LLC benefit and CPU cores have to pass through QPI to get data from remote memory for three times, which could incur performance drop dramatically (Fig. 1(c) red line). Nevertheless, if we place these two interdependent NFs in the same remote node, it can make subsequent NFs (i.e., NF5) get more high chance of cache hits and benefit the performance improvement from local packet memory access (Fig. 1 (d)). Moreover, the flow with the same sequence only needs to traverse the QPI for only twice.

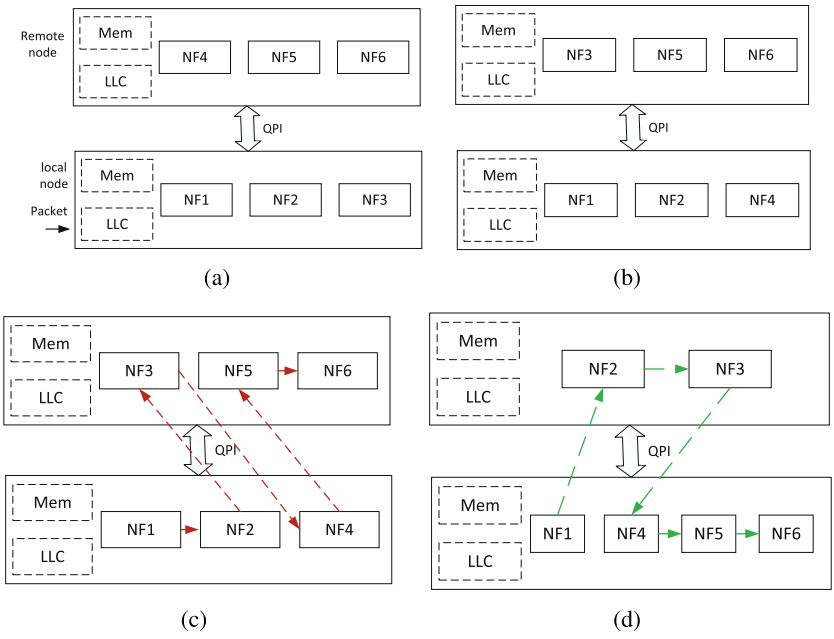

Fig. 1. Different placement for the same SFC. (Color figure online)

Therefore, we can see that more specific considerations are required in the placement of SFCs in the many-core system due to the dependency relation between NFs. The above observations motivate us to design an optimal strategy for the problem to achieve maximized aggregate throughput of all SFCs.

In this paper, we study the optimal placement of SFCs within a server. Compared with existing work in Sect. 2, we propose comprehensive solutions that consider the inequality of CPU cores in different nodes as well as different types

of NFs dependency relation. To understand the key factors in the optimization problem, we first formulate the problem with a Non-Linear Integer Programming (NLIP) model. Due to the NP-hardness of our problem, we then propose a heuristic-based backtracking placement algorithm to find an optimal or near-optimal solution in the range of reasonable time. Our design philosophy is to prevent existing NF couples from being placed on separated nodes as possible to avoid remote memory access and cache miss, but instead, let them be an entirety to be placed in the same node.

The remaining of this paper is organized as follows. Section 2 provides a brief overview of related work. We formulate the Interdependent VNFs Placement in Many-core System (IVPMS) problem in Sect. 3. Section 4 shows our proposed algorithm. The experiments and result are presented in Sect. 5. Section 6 gives a conclusion for our work.

2 Related Work

This section summarizes the state-of-art research on the placement of SFC in NFV.

As the promising domain for the next generation network infrastructure, NFV has attracted vast significant research attention. The institutions and enterprises are spending a huge amount of effort in this field. Most of the studies devoted to NFs and SFC placement in NFV. To the best of our knowledge, we are the first to formally study the SFC placement problem by combining the dependency relation between NFs among SFC with the characteristic NUMA architecture. We study and model performance affection by interdependency between NFs in many-core systems thoroughly and proposed an efficient algorithm to solve this problem.

There have been amounts of researches on NFs or SFCs placement in NFV. Rami et al. presented bi-criteria solutions for the actual placement of the virtual functions within the physical network [9]. It can reach constant approximation with respect to the overall performance and adhere to the capacity constraints of the network infrastructure. Defang et al. formulated the problem as an Integer Linear Programming (ILP) model and designed a two-stage heuristic solution to minimize the number of used physical machines [5]. [10] jointly considered the VNF placement and path section to utilize the network better. However, all the existing works above optimized SFC placement by mapping VNFs to the right servers. The server in this models is formulated as an entire node. They treat all CPU cores as equal without considering NUMA architecture in many-core systems. Thus, consideration without this factor will not be suitable and performance degradation will be incurred in actual situations.

Meanwhile, to make up for the insufficiency, many research efforts have been devoted to addressing the placement of SFC in a many-core system to achieve optimal execution performance. Zhilong et al. considered the inequality of CPU cores in respective NUMA node and proposed an NFV orchestrator to achieve maximum aggregate throughput of all SFCs in many-core systems [11]. Nevertheless, it overlooked the relationship of NFs amoung SFC. [12–15] relied on

frequent migration of threads to optimize thread-to-core placement to maximize task execution performance. However, comparing with threads, migrating NF progresses across cores or nodes brings significant performance overhead [16]. Besides, the heterogeneity of NFs could worsen the problem in NFV context and each running status of SFC contains a amout of flow information, resulting to the impossibility of migration. Moreover, to achieve high performance, advanced NFV systems bond NFs on dedicated CPU cores that cannot be scheduled by the operating system. This makes it difficult to migrate NFs among cores.

At the same time, it has been recognized as a challenge to place SFCs with certain dependency constraints between NFs. Wenrui et al. formulated the interdependent middleboxes placement as a graph optimization and proposed an efficient heuristic algorithm for the general scenario of a partially-ordered middlebox set [17]. [18] considered the dependency between NFs and presented a heuristic method to coordinate the composition of VNF chains and their embedding into the substrate network. Chaima et al. organized the VNFs placement in smaller interdependent subproblems and designed an efficient dynamic programming algorithm that runs in polynomial time [19]. Although considering the dependency within NFs, they did not in view of the fact that the deployment of same SFC on different core sets could result in significantly different throughput.

Different from the above ones, the formal model defined in this paper considers not only the NUMA architecture in modern CPU but also different types of NF dependency relations. Compared with the extensive studies on general SFC placement, we propose comprehensive solutions and conduct evaluations on hardware and software architectures.

3 Problem Formulation and Analysis

In this section, we present the system model and mathematical formulation of the IVPMS problem. We aim to find an optimized placement scheme for the SFCs, including NF couples in a many-core system.

The placement of SFCs is usually described as the assignment of a sequence of VNF instances to substrate nodes. Each node has limited available resources to host NF instances. To analyze the problem quantitatively and identify the key factors for problem-solving, we formulate the problem with the NLIP model and find a metric to measure the NF couple's effect formally.

3.1 Formulation of the Optimal Placement Problem

Formally, for many-core systems on the market, there are multiple nodes with an incremental number. Each node contains an identical number of CPU cores. We assume that there is a set of nodes V with C cores for each in a many-core system. A set of SFCs is denoted as S. For each SFC $s \in S$, let f_i^s be the i^{th} VNF, $1 \leq |i| \leq |s|$. Note that different SFCs may share the same VNF in practice. However, the path that traffic traverses can be considered separately. Moreover, flows tend not to arrive at the same time exactly, and multiple flows

are processed one by one [5]. Thus, we can overlook the impact and specify that chains in S are disjoint in our model.

We use $\leftarrow$ to describe the dependency relation between NFs. It is defined as a strict partial order. For $f_i, f_j \in s$, if $f_i \leftarrow f_j$, we say that the VNF f_j depends on VNF f_i. That's to say f_j should be chained after f_i. If $f \in s$ depends on no others, i.e.,$\forall f^{'} \in s, f^{'} \nleftarrow f$, we say that f has no interdependent VNF. Besides, the dependency relation has transitivity. If $f_i \leftarrow f_j$ and $f_j \leftarrow f_l$, then $f_i \leftarrow f_l$. We can see that a NF may has multiple interdependent NFs. For easy representation, we define $r\,(i,j) = 1$ if $f_i \leftarrow f_j$, and 0 otherwise.

In addition, we present a placement scheme, denoted as $P(s,i,k)$, to indicate whether NF i of chain s is located at the node k. It is a decisive binary variable defined as:

$$p(s,i,k) = \begin{cases} 1 \; if \; the \; i^{th} \; NF \; of \; chain \; \text{s} \; placed \; on \; node \; \text{k} \\ 0 \; otherwise \end{cases} \tag{1}$$

When all the NFs belonging to the same SFC s co-locate on the same node, the performance they achieve by sharing LLC and local memory access is the most optimal performance, which we set it as ϕ_s^{best}. We assume that the effect of contention can be ignored if the node has enough capacity to host all the NFs since NFs are stationed on specified cores. As involved in some work [4,10], the optimal performance can be measured by placing an entire SFC on the same node. Furthermore, when NFs included in the same SFC are co-located across nodes, the performance is referred to as attenuation performance and is defined as ϕ_s, of which $s \in S$. It's obvious that the attenuation performance is a referred value of optimal performance. We can see that the difference between the above two cases is cross-node placement.

The cross-node interdependent NFs flow traverses through are the vital key factor that affects the performance. Therefore, we introduce correlation index (denoted by $\varphi\,(i,j)$), which represents the effect of cross-node NF couple(i.e., (i, j)) to relate the optimal performances and attenuation performance. It should be noted that the objective is to maximize the aggregate performance across all required SFCs when deploying. Hence, we can formulate it as:

$$Max \sum_{s \in S} \phi_s \tag{2}$$

where:

$$\phi_s = \prod_{i,j \in s} \varphi\,(i,j)\,\phi_s^{best} \quad \forall i,j \in s, s \in S, \; r\,(i,j) = 1 \tag{3}$$

subject to the following constraints:

$$\left(\sum_{k \in V} P\,(s,j,k) * k - \sum_{k \in V} P\,(s,i,k) * k \right) * r\,(i,j) \geq 0 \quad if \; i \leftarrow j \tag{4}$$

$$\sum_{s \in S, i \in s} P(s,i,k) \leq C \quad \forall k \in V \tag{5}$$

$$\sum_{k \in V} P(s,i,k) = 1 \tag{6}$$

A brief explanation of this model is as follows. Equation (2) defines the optimization objective. Equation (3) describes the relationship between optimal performance, attenuation performance, and the effect of cross-node interdependent NFs. Constraints (4) enforces the dependency relation between NFs. In other words, flow must traverses i no later than j if the former is depended on by the latter. Equation (5) states that the resource consumed should not exceed the capacity of a many-core system during deployment. Equation (6) specifies that an NF should be deployed once and only once.

3.2 Formulation Analysis

The formulation of this problem gives us an intuitive description for finding optimal placement for S. However, we can see that the critical parameter, namely $\varphi(i,j)$ need to be figured out for problem-solving. Nevertheless, computing the correlation index is not-trivial due to some reasons.

As mentioned in Sect. 1, an SFC may contain multiple pairs of NF couples. If the NF couple co-locates in the same node, then there does not exist cross-node remote memory access through QPI, and the effect can be ignored (i.e., $\varphi(i,j) = 1$). Howeveer, for the estimation of placement, the prior knowledge we have are only the types and the chained sequence before acutal deployment. It's troublesome to adopt this information directly for the computation of the correlation index since they are not quantifiable. Of course, we can refer to some work [20] and manually analyze the system performance metric value to quantify the NF couple. However, metric value calculation could be platform depedent and burdensome.

Furthermore, this problem could be worse in the NFV context because NFs are usually heterogeneous and diverse, making the computation of $\varphi(i,j)$ are onerous. Moreover, there exists the contention of QPI resource if multiple pairs of interdependent NFs in the same SFC or among different SFCs adopt the scheme of cross-node placement. Besides, the chaining order for same SFC can change the count of the dependent NFs pair.

Consequently, we can see that more specific considerations are required for placing SFCs containing NF couples in many-core system due to above analysis.

Obviously, to address above problems, we should find some performance metrics to describe an NF and then describe the NF couple indirectly. Fortunately, there are many potential system-level metrics can be adopted [14,21]. Therefore, we should determine which one and automatically detect the appropriate metrics by existing tools. To achieve this goal, we first need to qualitatively analyze the characteristic these metrics should present.

Since the metrics should be able to describe an NF properly, its value should not vary with a large margin when located in different node. Besides, the metric value should be sensitive when NF change. That's to say, different NFs or same NF with different intrinsic properties, for instance, implementation or program languages, should have different metric values. Therefore, the metric values shoud be different when the performance are different for a set of NFs. In another word, the performance and metric value for an NF should exist strong correlation when NF run solely.

To figuratively express this correlation relationship, we assume P_i and m_i denote the ideal performance and metric value of NF i. Following this work [15,21], there exist linear correlation coefficient between the performance and metric value. Here we adopt mathematical Pearson Correlation Coefficient [22] to compute the correlation.

$$\rho\left(P_i, m_i\right) = \frac{Cov\left(P_i, m_i\right)}{\sigma_{P_i}\sigma_{m_i}} \tag{7}$$

$Cov\left(P_i, m_i\right)$ is the covariance for NF i between performance P_i and metric $m_[i$. σ_{P_i} and σ_{m_i} correspond to the varianes of performance and metric.

Then, we can deduce that the function of $\varphi\left(i, j\right)$ should be consistent with following function in form.

$$\varphi\left(i, j\right) = \frac{\alpha_i m_i \alpha_j m_j}{\alpha_i^{'} m_i^{'} \alpha_j^{'} m_j^{'}} \tag{8}$$

where $\alpha_i m_i$ and $\alpha_j m_j$ are the performance of NF couple i and j when they are located in the same node. α_i and α_j are the coefficient. Similarly, the denominator are the performance of NF couple i and j when they are placed across node. Besides, if assigning the processing order of the flow by NFs in advance, we can determine the sequence and the amount of NF couple. In simpler terms, we can get knowledge of the dependency relation inside a SFC.

As mentioned above, we cannot use limited existing knowledge as the input directly to expound the effect of interdependent NFs to IVPMS problem and the parameter is not known. Instead, we need to find some performance metrics to represent the effect of NF couple's cross-node placement formally. Undoubtedly, compared with local nodes, remote access through QPI in many-core system is the one of the main factors. Hence, we focus on the difference between local access and remote access across nodes. Remote access could cause LLC misses, QPI contention and so on. Single metric cannot profile all of factors clearly. Nevertheless, we find that the metric $resource_{stalls}$ profiled by OProfile [23] provides aggregate embodiment with the help of other research findings [11]. This metric including multiple sub indicators is appropriate to profile the aggregate performance. Consequently, we measure the metric of downstream NF among NF couples under above two cases. Then the function (i.e.,$\varphi\left(i, j\right)$) maps the metric value of NFs to attention performance (i.e.,ϕ_s).

To explore this function, we first check the relationship between the metric and performance. With more interdependent NF couples placed across nodes, $resource_{stalls}$ increases gradually and performance decreases, which shows negative correlation to remote access. Moreover, the function (i.e., $\varphi(i,j)$) shows positive correlation to local access if they are co-located in the same node. From this observation, we can approximately formulate this functions as:

$$\varphi(i,j) \approx \frac{resource\_stalls_{local}}{resource\_stalls_{remote}} \tag{9}$$

$$f(\varphi(1,2), \ldots \varphi(i,j) \ldots) = \alpha \prod_{i,j \in s} \varphi(i,j) + \beta \tag{10}$$

$$s.t.\ \alpha + \beta = 1 \tag{11}$$

Where α and β are tuning parameters to the aggregate effect of cross-node placement of interdependent NF couples to the SFC (the metric where NF couples placed in the local node is denoted as $resource\_stalls_{local}$ and other are relatively $resource\_stalls_{remote}$). Equation (10) states that if all of NFs (including NF couples) are co-located in the same node, the result of continued multiplication is equal to one, namely no effect to performance. Since multiple SFCs could be placed in all nodes, there should be distinct parameter groups due to the heterogeneity of chains. Thus, we define (α_s, β_s) for the case that chain s is deployed in the nodes.

We can see that if measuring samples are given, including the predetermined path of SFCs, accompanying the number of NF couple and the correlation index of each NF couple, it is easy to approximate the parameters (α, β) for each SFC. However, because of following reasons, we do not need to figure out them or find some mathematical method to approximate them. Instead, we propose a more efficient algorithm based on above analysis. One time-consuming but definite approach is to exhaust all possible schemes.

4 Backtracking Placement Algorithm

So far, we have constructed the concrete function formulation for correlation index between optimal performance and attenuation performance, which make IVPMS problem resolvable. However, it is not suitable for practical deployment. As can be seen, its time complexity is $O\left(K^C\right)$ (K is the number of available nodes and C is the number of total NFs), which is not satisfied the floor threshold of response time for operator. Furthermore, the relaxed model of IVPMS problem is NP-hard [17]. Therefore, there is no polynomial-time approximation algorithm to find an optimal placement. A alternative approach to finding optimal solution is brute-force search. Although it must can find the optimal placement, the time complexity is same with abvoce, which cannot tolerable for practical application. Thus, we propose a heuristic-based backtracking placement algorithm (BPA), to find the optimal or near-optimal solution.

The basic idea behind our heuristic algorithm is to place all of NF couples in the same node as possible while minimizing the number of remote access across nodes. Instead of finding placement for each NF, we first treat all of NFs in the same SFC as entirety and place an entire chain on the same node. It is based on an important empirical observation that the performance will drop dramatically as the number of cross-node placement event in the chain increase. (This is because cross-node access bring the augment of $resouce\_stalls_{remote}$, minishing the value of $\varphi(i,j)$). Then, we move NFs selectively to remote node due to resource constraints. The selection of NFs following two priority principle: (1) search NFs that have nothing to do with position in chain and remove them to remote nodes. If this NFs have no interdependent relation with other NFs inside chain, they can be placed in an arbitrary order; (2) remove NF couples at the end of chain to remote nodes. After removing NFs without dependency to remote node, the remains are completely-dependent and totally-ordered. If the local node cannot hold the rest of NFs yet, we still need to remove subchain in the tail of SFC from local node to remote node. The split point in the rest chain to form a subchain is the NF among NF couples with least performance degradation, denoted as LPD-NF (i.e., $\varphi(i,j)$ with the highest value).

The pseudo-code of placement algorithm is shown in Algorithm 1. It first places all of SFCs in the same local node without considering the resource limitation. In the following steps, the main work is to remove the out-of-limit NFs to remote node. In such a case, we choose NFs based on above two principles. We search the ones that no dependency exists between NFs and remove them to remote node in priority. Furthermore, we iterate each SFC to move subchain from local node to remote node. We adopt deep-first search (DFS) to find the LPD-NF for each chain and update the potential optimal placement, meanwhile examine whether the capacity of local node could cover the requirement. Once reaching the lower bound, we will terminate this iteration. Finally, we will find the potential optimal solution with high performance in all possible placements.

5 Performance Evaluation

In this paper, we use a high performance NFV platform, OpenNetVM [4] to run NFs and SFCs. We select several type of NFs for evaluation, including IPv4 Router (Rout), Firewall (FW), deep packet inspection (DPI) and Switch (SW). We run our algorithm and OpenNetVM on the same server. The computer is a two-NUMA-node server, which is equipped with Intel(R) Xeon(R) Silver 4114 CPU, 256G RAM and two dual-port DPDK compatible NICs located on node 0. The operation system server runs is Centos 7.2 with kernel version 3.10.0-693.el7.x86 64. The input traffic is generated by the DPDK Pktgen tool [24].

Before proceeding with our evaluation, we need to measure the metric value of each NF. To facilitate the computation of $\varphi(i,j)$ and speed up the search of split point, we assume that there is dependency relation between each other. To demonstrate the correlation index, we measure the metric value under two different deployment situation, namely co-locating placement or cross-node placement. We measure the metric for many times and finally get the average result.

Algorithm 1: Backtracking placement algorithm

Input: S:SFC sets, C: the number of cores in a node
Output: O: Placement of each NF in the SFCs

```
1  //init placement
2  O_init ← place all of SFCs on the local node
3  Max_perf ← 0 //Optimal performance
4  Sort all of SFCs based on φ_s^best in ascending order
5  while |s_local| ≥ C do
6      (s_local, s_remote) ← initialization // |s_remote| = 0
7      r_c = C
8      foreach s ∈ S do
9          NF_s ← find NFs that without dependency
10         if |NF_s| ≤ r_c then
11             (s_local, s_remote) ← (s_local − NF_s, s_remote + NF_s)
12             r_c ← r_c − |NF_s|
13             if |s_local| ≤ C then
14                 Set flag
15         UpdatePlacement(Max_perf,s,NF_s)
16     if flag then
17         break
18     DFS(s_local,s_remote,S,r_c,Max_perf)
19 Function DFS(s_local,s_remote,S,r_c,perf):
20     foreach s ∈ S do
21         LPD_NF ← find split point with maxφ(i, j)
22         s_subchain ← (LPD_NF : NF_tail)
23         if r_c > |s_subchain| then
24             (s_local, s_remote) ← (s_local − s_subchain, s_remote + s_subchain)
25             r_c ← r_c − |s_sunchain|
26         UpdatePlacement(perf,s,s_subchain)
27         if |s_local| ≤ C or |s| = 0 then
28             continue
29         DFS(s_local,s_remote,S,r_c,perf)
30 Function UpdatePlacement(perf,s,NF_s):
31     if perf < Perf_measured then
32         perf ← Perf_measured
33         s ← s − NF_s
```

Figure 2(a) shows the average metric value for each type of NF. We can see that when multiple NFs are co-located in the same node, the metric value varies with small margin. Figure 2(b) shows the average correlation index for different NF couple. By this way, we can reduce the search time enormously.

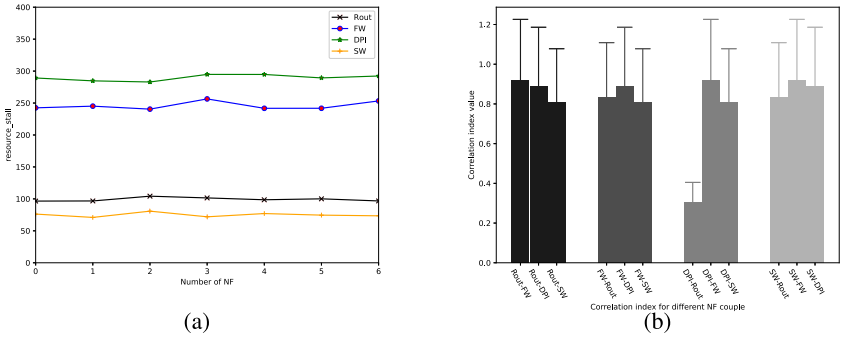

Fig. 2. Metric value for NFs and NF couples.

To demonstrate that our heuristic algorithm can improve the aggregate performance by finding optimal or near-optimal placement solution, we also implement two placement strategies: (1) greedy placement(GP): it co-locates the NFs on the local node in priority until exhaust the cores, then places the remains on the other nodes; and (2) fairness placement(FP); it attempts to place SFCs on the all nodes in sequence. According to [25,26], the length of currently deployed service chains do not exceed seven. Thus, the number of VNFs in a chain can range from 2 to 7. In this paper, we use 5 sets of SFC. Each set includes 3 SFCs, which consist of above NFs.

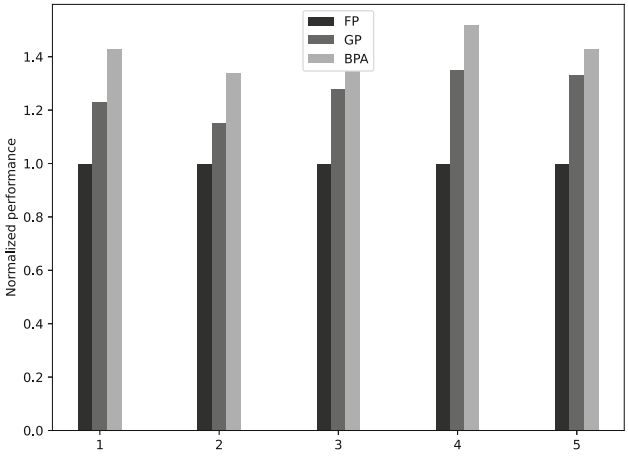

Fig. 3. Aggregate performance for three placement schemes under different number of SFC sets.

Figure 3 shows the normalized performance of above three placement algorithms. We can see that in all SFC sets, our algorithm can achieve the best aggregate performance among three placement strategies. Comparing with greedy

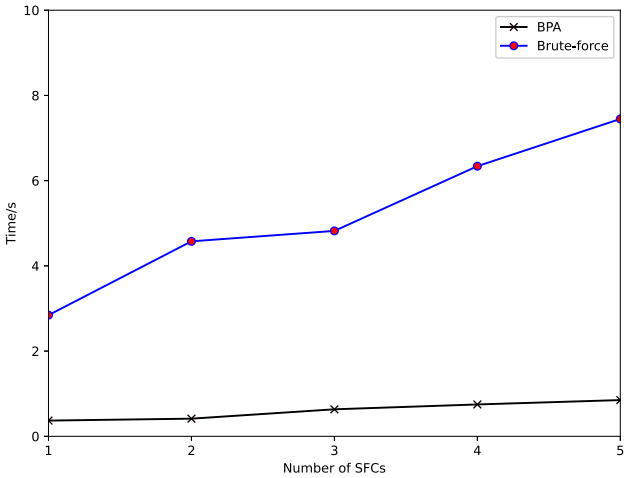

Fig. 4. Time cost under different numbers SFC sets.

placement, BPA improves the aggregate performance by an average of about 46%. Also, BPA improves the aggregate performance by an average of about 20% compared with fairness placement.

We also evaluation the efficiency of our algorithm. We compare the backtracking placement algorithm with the brute-force method. We randomly select two sets from above SFC sets as input and increase the number of placed SFC gradually. We repeat the experiment for multiple times and record the cost time. Figure 4 shows that our algorithm can find a solution less than two second, even in the worst case with five SFCs. Comparing with brute-force search strategy, our algorithm can reduce the time to find the near-optimal placement tremendously.

6 Conclusion

In this paper, we study the SFCs placement in a many-core system to improve the aggregate performance. In order to address the placement problem, we formulate it with a NILP model. Besides, to reveal the relation between optimal performance and attention performance, we find a reasonable metric to weigh the effectiveness of NF couple systematically and describe the relation with formulation. Furthermore, we present a heuristic-based backtracking placement algorithm to search optimal or near-optimal placement for SFC in a many-core system. Our evaluation built on OpenNetVM shows that BPA can improve the aggregate performance significantly and has high efficiency to find optimal or near-optimal solutions.

References

1. Han, B., Gopalakrishnan, V., Ji, L., Lee, S.: Network function virtualization: challenges and opportunities for innovations. IEEE Commun. Mag. **53**(2), 90–97 (2015)
2. Mehraghdam, S., Keller, M., Karl, H.: Specifying and placing chains of virtual network functions. In: 2014 IEEE 3rd International Conference on Cloud Networking (CloudNet), pp. 7–13. IEEE (2014)
3. Palkar, S., et al.: E2: a framework for NFV applications. In: Proceedings of the 25th Symposium on Operating Systems Principles, pp. 121–136. ACM (2015)
4. Zhang, W., et al.: OpenNetVM: a platform for high performance network service chains. In: Proceedings of the 2016 Workshop on Hot Topics in Middleboxes and Network Function Virtualization, pp. 26–31. ACM (2016)
5. Li, D., Hong, P., Xue, K., et al.: Virtual network function placement considering resource optimization and SFC requests in cloud datacenter. IEEE Trans. Parallel Distrib. Syst. **29**(7), 1664–1677 (2018)
6. Sefraoui, O., Aissaoui, M., Eleuldj, M.: OpenStack: toward an open-source solution for cloud computing. Int. J. Comput. Appl. **55**(3), 38–42 (2012)
7. Cisco: Cisco: Nat order of operation (2014). http://www.cisco.com/c/en/us/support/docs/ip/network-address-translation-nat/6209-5.html
8. TechNet, M.: Microsoft technet: VPNs and firewalls (2015). https://technet.microsoft.com/en-us/library/cc958037.aspx
9. Cohen, R., Lewin-Eytan, L., Naor, J.S., Raz, D.: Near optimal placement of virtual network functions. In: 2015 IEEE Conference on Computer Communications (INFOCOM), pp. 1346–1354. IEEE (2015)
10. Kuo, T.W., Liou, B.H., Lin, K.C.J., Tsai, M.J.: Deploying chains of virtual network functions: on the relation between link and server usage. IEEE/ACM Trans. Netw. (TON) **26**(4), 1562–1576 (2018)
11. Zheng, Z., et al.: Octans: optimal placement of service function chains in many-core systems. In: IEEE Conference on Computer Communications, IEEE INFOCOM 2019, pp. 307–315. IEEE (2019)
12. Zhuravlev, S., Blagodurov, S., Fedorova, A.: Addressing shared resource contention in multicore processors via scheduling. ACM Sigplan Not. **45**, 129–142 (2010)
13. Lepers, B., Quema, V., Fedorova, A.: Thread and memory placement on NUMA systems: asymmetry matters, pp. 277–289 (2015)
14. Rao, J., Wang, K., Zhou, X., Xu, C.: Optimizing virtual machine scheduling in NUMA multicore systems, pp. 306–317 (2013)
15. Liu, M., Li, T.: Optimizing virtual machine consolidation performance on NUMA server architecture for cloud workloads. In: International Symposium on Computer Architecture, vol. 42, no. 3, pp. 325–336 (2014)
16. Gemberjacobson, A., et al.: OpenNF: enabling innovation in network function control. ACM Spec. Interest Group Data Commun. **44**(4), 163–174 (2015)
17. Ma, W., Sandoval, O., Beltran, J., Pan, D., Pissinou, N.: Traffic aware placement of interdependent NFV middleboxes. In: IEEE Conference on Computer Communications, IEEE INFOCOM 2017, pp. 1–9. IEEE (2017)
18. Beck, M.T., Botero, J.F.: Coordinated allocation of service function chains, pp. 1–6 (2014)
19. Ghribi, C., Mechtri, M., Zeghlache, D.: A dynamic programming algorithm for joint VNF placement and chaining. In: Proceedings of the 2016 ACM Workshop on Cloud-Assisted Networking, pp. 19–24. ACM (2016)

20. Dobrescu, M., Argyraki, K., Ratnasamy, S.: Toward predictable performance in software packet-processing platforms, p. 11 (2012)
21. Nathuji, R., Kansal, A., Ghaffarkhah, A.: Q-clouds: managing performance interference effects for QoS-aware clouds, pp. 237–250 (2010)
22. Statistics solutions: Pearson's correlation coefficient (2015). https://www.statisticssolutions.com/pearsons-correlation-coefficient/
23. Sourceforge: Oprofile (2018). https://oprofile.sourceforge.io/news/
24. Online: DPDK pktgen (2019). http://pktgen-dpdk.readthedocs.io/en/latest/
25. Haeffner, W., Napper, J., Stiemerling, M., Lopez, D., Uttaro, J.: Service function chaining use cases in mobile networks. Internet Engineering Task Force (2015)
26. Kumar, S., Tufail, M., Majee, S., Captari, C., Homma, S.: Service function chaining use cases in data centers. IETF SFC WG 10 (2015)