Clustering-based Allocation of Virtualized Functions for Service Function Chaining

Hidehiro Kanemitsu1,2, Masaki Hanada3, and Hidenori Nakazato4

1 School of Computer Science, Tokyo University of Technology, Japan.
2 Waseda Research Institute for Science and Engineering, Waseda University, Japan.
3 Department of Information Systems, Tokyo University of Information Sciences, Japan.
4 Department of Communications and Computer Engineering, Waseda University, Japan.
E-mail: kanemitsu@stf.teu.ac.jp.

Abstract

Network Function Virtualization (NFV) has been widely spread for software-level network configuration. Such functionalities interact each other to achieve data-flow processing, named as Service Function Chaining (SFC). In SFC, each execution unit, named as Virtualized Network Function (VNF), has dependencies with others. Such configuration is realized by allocating each VNF to a VM. Conventional approaches mainly focus on how to allocate each VNF to a VM, or how to select a VNF and they are resolved separately. In this paper, we propose a VNF clustering algorithm, called Clustering for Minimizing Worst Schedule Length for VNF (CMWSL-VNF) to resolve both VNF allocation and VNF selection simultaneously. CMWSL-VNF can greatly reduce the data communication latency and maintains the degree of parallelism by the clustering algorithm. Experimental results show that CMWSL-VNF can utilize vCPUs to minimize the response time.

1. Introduction

The communication network has been more and more utilized to support our daily lives. Recent advancement in communication network involves virtualization for each network equipment, such as router, firewall, load balancer, and so on. Such network equipment can be virtualized to achieve portability as a software level, i.e., network virtualization. However, such situation arises a new issue for the network optimization. For instance, how each virtualized function should be deployed, and how each virtualized function should be selected from the set of VMs in order to minimize the latency, the response time, or to maximize the throughput are critical problems for network virtualization. In particular, each virtualized network function, called VNF, has dependencies with others, thereby each VNF composes a Service Function Chaining (SFC). SFC allows every network function to be virtualized and then each virtualized function can have the portability. Such characteristic is very useful when network administrators must consider how to deploy network device, because no physical network device is needed by SFC.

However, SFC has several performance issues, e.g., how to minimize the communication latency among VNFs, how to minimize the number of VN instances to be allocated for VNFs, and how to minimize the response time, and so on. Conventional approaches for resolving those issues focus on optimization problem by formulating the model. As a result, such approach has very high time complexity and therefore they are not able to be applied for a practical use.

In this paper, we propose a VNF clustering algorithm, called Clustering for Minimizing Worst Schedule Length for VNF (CMWSL-VNF), to deriving a solution for VNF allocation and VNF selection. Each VNF is clustered if WSL is made smaller by a clustering, thereby WSL is minimized. Then we present that minimizing WSL can contribute to minimize the schedule length by the experimental results.

2. System Model

2.1 Network Model

The set of node, having multiple VMs in the network is defined as \( N = \{ n_1, n_2, \ldots \} \), and let each VM be \( M_i = \{ m_{i,1}, m_{i,2}, \ldots \} \), where \( n_i \in N \). The processing speed (MIPS) and the communication bandwidth of \( m_{i,j} \) are \( \alpha_{i,j} \) and \( \beta_{i,j} \), respectively. For each VM \( m_{k,l} \in n_k \), \( m_{k,l} \) has two vCPUs, i.e., one virtual CPU corresponds to one logical CPU, which is defined as \( C_{vcpu} \).

2.2 SFC and VNF

There are one SFC, defined as \( S = (V, E) \), where \( V \) is the set of VNFs, and \( E \) is the edges among VNFs, i.e., the data dependencies. The \( i \)-th VNF is \( v_i \in V \), and the data dependency from \( v_i \) to \( v_j \) is expressed as \( e_{i,j} \in E \). \( w_i \) is the workload of \( v_i \), and \( c_{i,j} \) is the data size of \( e_{i,j} \). In a workflow,
a VNF cannot begin execution until all data from its predecessor VNFs arrive. Let \( \text{pred}(v_i) \) be the set of predecessor VNFs of \( v_i \), and let \( \text{suc}(v_i) \) be the set of successor VNFs of \( v_i \). If \( \text{pred}(v_i) = \emptyset \), \( v_i \) is called the START VNF, and if \( \text{suc}(v_i) = \emptyset \), \( v_i \) is called the END VNF.

### 2.3 Cost Model

Let \( v_i \) be the workload of \( v_i \) and the resultant data size of \( v_i \), i.e., that must be sent from \( v_i \) to \( v_j \) is defined as \( d_{i,j} \). The processing time of \( v_i \) on \( c_{k,l,m} \) is defined as \( T_p(v_i, c_{k,l,m}) = \frac{w_i}{\alpha_{k,l,m}} \), and the communication time of \( d_{i,j} \) from \( m_{k,l} \) to \( m_{p,q} \) is defined as

\[
T_c(d_{i,j}, L_{k,p}) = \frac{d_{i,j}}{\min\{\beta_k, \beta_p\} L_{k,p}},
\]

where \( \beta_k \) is the communication bandwidth of \( n_k \).

### 2.4 Objective Function

Let the assignment target of \( v_i \) be \( A(v_i) \), where \( A(v_i) \) is a vCPU. Here, we present how the start time of each VNF \( v_i \) on \( A(v_i) \), i.e., \( T_s(v_i, A(v_i)) \) is determined when scheduling VNFs and how the schedule length is derived. Suppose the finish time of \( v_i \) is defined as \( T_f(v_i, A(v_i)) \), then we have

\[
T_f(v_i, A(v_i)) = T_s(v_i, A(v_i)) + T_p(v_i, A(v_i)),
\]

where \( A(v_i) \in M \cup \text{VCPU} \). The set of free VNFs is defined as \( f\text{List} \); that is, the set of VNFs whose predecessor VNFs have been scheduled. For each VNF in \( f\text{List} \), the data ready time can be derived. Data ready time (DRT) is the maximum data arrival time from all predecessor VNFs. Note that DRT is the earliest start time for each VNF and the actual start time can be later than DRT. This is because \( v_j \) cannot start execution when a VNF \( v_j \notin \text{pred}(v_i) \) such that \( A(v_i) = A(v_j) \) is scheduled before \( v_j \). DRT of \( v_j \) at \( m_k \) is derived as follows:

\[
T_{dr}(v_j) = \max_{v_i \in \text{pred}(v_j)} \{T_f(v_i, A(v_i)) + T_c(d_{i,j}, L_{k,p})\},
\]

where \( T_c(d_{i,j}, L_{k,p}) = 0 \) if \( k = p \), where \( A(v_i) \in n_k \) and \( A(v_i) \in n_p \).

Then we can derive the start time, i.e., \( T_s(v_j, A(v_j)) \) with DRT as follows:

\[
T_s(v_j, A(v_j)) = \max_{v_i \notin \text{pred}(v_j)} \{T_f(v_i, A(v_i)), T_{dr}(v_j)\},
\]

The schedule length is the finish time of the END VNF and is defined as follows:

\[
T_f(v_{end}, A(v_{end})) = T_s(v_{end}, A(v_{end})) + T_p(v_{end}, A(v_{end})).
\]

With aforementioned definitions, the objective function is defined as follows:

#### Algorithm 1: CMWSL-VNF Algorithm

**Input:** Set of vCPUs \( \text{VCPU} \) and set of VNF clusters \( \text{CLS} = V \).

**Output:** The mapping from \( \text{cls}(i) \in \text{CLS} \) to \( \text{VCPU} \) and the schedule.

1. \( UEX = \emptyset \) is the set of unclustered VNF clusters and \( FREE \) is the set of VNF clusters which are ready for clustering.
2. \( UEX = \emptyset \) \( UEX \neq \emptyset \) do
3. Select \( \text{cls}(i) \) having the maximum \( \text{WSL} \) in \( FREE \) using the average proc. speed \( v_{cpu}(v) \), and the average comm. bandwidth \( \beta_{vnf} \)
   - Suppose \( v_i \in \text{top}(i) \) dominates \( \text{TL}(i) \) and \( v_j \in \text{hbm}(i) \) dominates \( \text{BL}(i) \).
   - Suppose \( v_i \in \text{pred}(v_i) \) dominates \( \text{Hello}(v_i) \) and \( v_i \in \text{suc}(v_i) \) dominates \( \text{Hello}(v_i) \) and \( v_i \in \text{cls}(h) \) and \( v_i \in \text{cls}(j) \).
4. if \((8)\) is satisfied with \( \text{cls}(i) \) then
   - \( /\*\) Downward clustering. \( /\*\)
   - \( UEX = UEX - \{\text{cls}(i)\} \).
   - \( \text{cls}(i) \rightarrow \text{cls}(i) \cup \text{cls}(j) \).
5. else if \((9)\) is satisfied with \( \text{cls}(i) \) then
   - \( /\*\) Upward clustering. \( /\*\)
   - \( UEX = UEX - \{\text{cls}(h)\} \cup \text{cls}(i) \).
   - \( \text{cls}(i) \rightarrow \text{cls}(h) \cup \text{cls}(i) \).
6. else
   - \( /\*\) End of clustering for \( \text{cls}(i) \).
   - \( \text{FREE} \rightarrow \text{FREE} - \{\text{cls}(i)\} \).
   - Put set of clusters \( \text{clus}(j) \) satisfying \((10)\) are put into \( \text{FREE} \) by tracing successor VNF of \( \text{out}(i) \).
   - Select \( q_{i,l,m} \in \text{CLS} \) satisfying \((11)\) and \( \text{cls}(i) \) is assigned to \( q_{i,l,m} \).
7. Update \( \text{Hello}(v_i) \) and \( \text{blewel}(v_i) \) for each \( v_i \in V \) using actual assigned vCPUs.
8. \( /\*\) VNF ordering \( /\*\)
9. Let \( UEX_{onf} \rightarrow V \), \( \text{FREE}_{onf} \rightarrow \text{START VNFs}\).
10. while \( UEX_{onf} \neq \emptyset \) do
11. Select the VNF \( v_i \) satisfying \((13)\) from \( \text{FREE}_{onf} \).
12. Insert the time slot on \( A(v_i) \) by which \((12)\) is satisfied and the \( T_f(v_i, A(v_i)) \) is minimized.
13. \( \text{FREE}_{onf} = \text{FREE}_{onf} - \{v_i\} \); \( \text{UEX}_{onf} = \text{UEX}_{onf} - \{v_i\} \).
14. return \( V \) and \( \text{VCPU} \).

**Objective Function 1** Minimize \( T_f(v_{end}, A(v_{end})) \).

### 3. CMWSL-VNF Algorithm

#### 3.1 Assumptions

CMWSL-VNF consists of two phases, i.e., (i) VNF clustering phase and (ii) VNF ordering phase. At (i), each VNF belongs to a “single VNF cluster” having only one VNF. Then the two VNF clusters are merged into the larger VNF cluster in order to minimize the possible schedule length. Since the actual schedule length cannot be determined until each VNF is allocated to a time slot of a vCPU, at (i) only a “measure” for the schedule length is derived. We name the measure as “Worst Schedule Length (WSL)”. The objective of (i) is to minimize WSL. A VNF cluster includes one or more VNFs, and let define the i-th VNF cluster as \( \text{cls}(i) \), where the initial cluster \( \text{cls}(i) = \{v_i\} \), and we define “clustering \( \text{cls}(i) \) and \( \text{cls}(j) \)” as \( \text{cls}(i) \rightarrow \text{cls}(i) \cup \text{cls}(j) \). For a VNF cluster \( i \), \( \text{top}(i) \) is the set of VNFs that begin executing first in \( \text{cls}(i) \), \( \text{in}(i) \) is the set of VNFs with incoming edges from other VNF clusters, and \( \text{out}(i) \) is the set of VNFs with outgoing edges to
other VNF clusters; \textit{btm}(i) is the set of VNFs that has no immediate successor VNFs in \textit{cls}(i). WSL of \textit{cls}(i) is defined as
\[
WSL(\textit{cls}(i)) = TL(\textit{cls}(i)) + T_p(\textit{cls}(i), A(\textit{cls}(i))) + BL(\textit{cls}(i)),
\]
where \(A(\textit{cls}(i)) \in n_p\) and
\[
TL(\textit{cls}(i)) = \max_{v_k \in \text{top}(i)} \{t_{\text{level}}(v_k)\},
\[
t_{\text{level}}(v_k) = \max_{v_j \in \text{pred}(v_k)} \{t_{\text{level}}(v_j) + T_p(v_j, A(v_j)) \} + T_c(d_{j,k}, L_{p,q}),
\]
\[
BL(\textit{cls}(i)) = \max_{v_k \in \text{btm}(i)} \{t_{\text{level}}(v_k)\},
\]
\[
b_{\text{level}}(n_k) = \max_{v_j \in \text{in}(n_k)} \{T_p(v_k, A(v_k)) + T_c(d_{k,l}, L_{q,r}) + b_{\text{level}}(v_l)\},
\]
where we suppose that \(v_j \in \textit{cls}(h), v_k \in \textit{cls}(i), v_l \in \textit{cls}(h), \)
and \(A(\textit{cls}(h)) \in n_p, A(\textit{cls}(i)) \in n_q, \) \(A(\textit{cls}(j)) \in n_r, \)
and \(\text{desc}(v_j, \textit{cls}(k))\) is the set of descendant VNFs of \(v_j\) in \(\textit{cls}(k)\), i.e., at \(t_{\text{level}}(v_j)\) at (7) is the possible latest start time of \(v_j\), provided that \(v_j\) is scheduled as late as possible when every VNF in \(\textit{in}(j)\) can start execution without waiting the data arrival. Then the vCPU, by which \(W SL(\textit{cls}(i))\) is minimized, is selected for allocating \(\textit{cls}(i)\). We assume that each vCPU has at most one VNF cluster, otherwise the finish time of a VNF can be delayed by multiple VNF cluster allocation.

### 3.2 VNF Clustering and vCPU Allocation

First, suppose that every VNF cluster has only one VNF, and each VNF cluster belong to \(\textit{UEX}\), the set of unexamined VNF clusters. If a VNF cluster is clustered, it is removed from \(\textit{UEX}\) and the algorithm finishes when \(\textit{UEX} = \emptyset\). CMWSL-VNF tries to select the VNF cluster as “pivot” having the maximum WSL at (6) from the free VNF cluster list \(\textit{FREE}\). From the pivot, the successor VNF cluster or predecessor VNF cluster is selected as a clustering target. Let name the clustering from \(\textit{pivot}\) to a successor VNF cluster as "downward clustering", while the clustering from \(\textit{pivot}\) to a predecessor VNF cluster as "upward clustering". The condition that downward clustering is accepted is as follows:
\[
TL(\textit{cls}(i)) + T_p(\textit{cls}(i) \cup \textit{cls}(j), \alpha_{ave}) + BL(\textit{cls}(j)) \leq WSL(\textit{cls}(i)),
\]
where \(v_b \in \text{btm}(i)\) dominates \(BL(\textit{i})\), and \(v_c \in \text{succ}(v_b)\) dominates \(b_{\text{level}}(v_c)\), \(i \in \textit{cls}(j)\). If downward clustering is not accepted, the algorithm tries to perform upward clustering.

The condition that upward clustering is accepted is defined as follows:
\[
TL(\textit{cls}(h)) + T_p(\textit{cls}(h) \cup \textit{cls}(i), \alpha_{ave}) + BL(\textit{cls}(i)) \leq WSL(\textit{cls}(i)),
\]
where \(v_t \in \text{top}(i)\) dominates \(TL(\textit{i})\), and \(v_s \in \text{pred}(v_t)\) dominates \(t_{\text{level}}(v_t)\), \(s \in \textit{cls}(h)\). If either one condition is satisfied, clustered VNF clusters are removed from \(\textit{UEX}\). If both (8) and (9) are not satisfied, the pivot remains unclustered and then the new cluster \(\textit{cls}(j)\) satisfying the following condition is put into \(\textit{FREE}\). A VNF cluster satisfying the following condition is put into \(\textit{FREE}\).
\[
\{\textit{cls}(j) | \forall\textit{cls}(i) \in \textit{UEX} s.t. \text{ } v_s \in \textit{cls}(i), v_s \in \text{pred}(v_t), v_t \in \text{top}(i)\}.
\]
At \(\textit{cls}(j)\), all predecessor VNFs of \(\text{top}(j)\) are supposed to belong to traced VNF clusters. In such a case, \(\textit{cls}(j)\) is put into \(\textit{FREE}\).

When a pivot cannot be clustered, the assignment target of vCPU is determined. VCPU scheduling the following condition is selected for allocating pivot (let pivot be \(\textit{cls}(i)\)).
\[
c_{q,t,m} \in C_{\text{vcpu}} s.t \min_{o(c_{q,t,m})=0 \{T_c(d_{i,j}, L_{p,q}) + T_p(\textit{cls}(i), c_{q,t,m}) + T_c(d_{s,t}, \beta_{q})\}}
\]
where \(v_t \in \textit{cls}(h), v_t \in \text{top}(i)\) dominates \(t_{\text{level}}(v_t)\), \(v_s \in \text{pred}(v_t)\) dominates \(TL(\textit{cls}(i))\). Also \(t_s\) dominates \(BL(\textit{cls}(i))\) and \(v_t\) dominates \(b_{\text{level}}(v_s)\). Algorithm 1 show the overall procedures. Lines 1 to 13 in Algorithm 1 correspond to the VNF clustering and vCPU allocation phase.

### 3.3 VNF Ordering

Since each VNF is allocated to a vCPU by the VNF clustering and vCPU allocation in the previous section, it can be ordered for the actual scheduling. As can be seen at line 15 in Algorithm 1, let define \(\textit{UEX}_{vnf} = \emptyset\) and \(\textit{FREE}_{vnf}\). In this VNF ordering phase, the objective is to assign each VNF to a time slot of the allocated vCPU in order not to exceed the pre-defined load of the CPU core, because excessive VNF allocation to a vCPU cause a serious performance degradation due to vCPU stacking and lock holder preemption among vCPUs. When the selected VNF is selected and allocated to a time slot on the vCPU \(c_{k,l,m}\), whether the average load per vCPU exceeds the threshold load of the core \(c_{k,l}\) or not should be taken into account. If a cloud, overcommitment in terms of vCPU must satisfy the following condition:
\[
\frac{1}{N_{\text{vcpu}}} \sum_{m=1}^{2} W L_{\text{t}}(c_{k,l}) \geq 1,
\]
where \(W L_{\text{t}}(c_{k,l})\) is the threshold of the load of the CPU core \(c_{k,l}\), \(N_{\text{vcpu}}\) is the number of vCPUs per core, and
Table 1: Simulation Parameters

<table>
<thead>
<tr>
<th>VNF</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Out degree of VNF</td>
<td>1 to 5</td>
<td></td>
</tr>
<tr>
<td>Workload of VNF</td>
<td>500 to 100000 (MI)</td>
<td></td>
</tr>
<tr>
<td>Data size among VNFs</td>
<td>1MB to 1GB</td>
<td></td>
</tr>
<tr>
<td>Load for each VNF</td>
<td>10% to 90%</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>System</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number of cores / node</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Number of vCPUs / core</td>
<td>2 to 18</td>
</tr>
<tr>
<td></td>
<td>Threshold load / core</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Comm. Bandwidth at node</td>
<td>70%</td>
</tr>
<tr>
<td></td>
<td>MIPS at vCPU</td>
<td>100 MB to 1GB</td>
</tr>
<tr>
<td></td>
<td>2000 to 3000 (mips)</td>
<td></td>
</tr>
</tbody>
</table>

\[ WL(c_{k,l,m}) \] is the load of \( c_{k,l,m} \). That is, (12) means that the average load for each vCPU must not exceed the threshold load of the core for achieving overcommitment.

In the VNF ordering phase, the VNF \( v_i \in FREE_{vnf} \), satisfying the followin condition is selected from \( FREE_{vnf} \).

\[
 tlevel(v_i) + blevel(v_i) = \max_{v_k \in FREE_{vnf}} \{ tlevel(v_k) + blevel(v_k) \} \tag{13}
\]

Then a \( v_i \) is assigned to a time slot in the vCPU \( c_{k,l,m} \) and then (12) is still satisfied, the assignment is accepted. This VNF ordering phase corresponds to lines 15 to 19 in Algorithm 1.

4. Experimental Results

4.1 Simulation Setup

We conducted the comparison by the simulation, which was developed on CloudSim[1] with incorporating WorkflowSim[2]. The development and running platform are jdk1.7.0.51, and CPU is Intel(R) Core i7-5600U 2.6 GHz with 8 GB RAM. Table 1 show the parameter set used in the simulation.

4.2 Results

We compared the schedule length and the number of allocated vCPUs with CoordVNF[3] and HEFT-based SFC[4]. Both the schedule length and the number of vCPUs are averaged through 100 run, where the size of SFC, i.e., \( |V| \) is varied for each run. Table 2 show the comparison results, where values corresponds to the ratio to that by CMWSDL-VNF. We observe that CMWSDL-VNF outperforms other two approaches in terms of both the schedule length and the number of vCPUs, while HEFT-based SFC provides the best running time for its algorithm simplicity. However, the running time by CMWSDL-VNF is acceptable level but the running time by CoordVNF is very long because it is based on a optimization approach.

5. Conclusion

6. Acknowledgement

This work is partially supported by the R&D contract “Wired-and-Wireless Converged Radio Access Network for Massive IoT Traffic” with the Ministry of Internal Affairs and Communications, Japan, for radio resource enhancement. This work is also supported by the EU-JAPAN initiative by the EC Horizon 2020 Work Programme (2018-2020) Grant Agreement No.814918 and Ministry of Internal Affairs and Communications “Federating IoT and cloud infrastructures to provide scalable and interoperable Smart Cities applications, by introducing novel IoT virtualization technologies (Fed4IoT)”.

References


