Multi-Cloud Distribution of Virtual Functions and Dynamic Service Deployment: OpenADN Perspective

Deval Bhamare, Qatar University
Raj Jain, Washington University in St. Louis
Mohammed Samaka, Qatar University
Gabor Vaszkun, Washington University in St. Louis
Aiman Erbad, Qatar University

2nd IEEE International workshop on Software Defined Systems
Tempe, AZ, 9th March, 2015

These slides and audio/video recordings of this talk are at:
http://www.cse.wustl.edu/~jain/talks/vm_distp.htm
Overview

1. Virtual Functions
2. Optimal and non-optimal location
3. Integer Linear Programming Formulation
4. Minimum Residue Algorithm
5. Comparison with Max-Min Algorithm
Virtual Functions

- **Virtual Functions**: Distributed entities: e.g., Web server, load balancers, data base servers, Proxy servers and many others
Service Chaining

- Individual Virtual Functions communicate with each other in defined ways
- Traffic between components is required (by policy) to flow through specialized network services (e.g., firewalls, IDS, etc.)

Source: IETF-87
VFs, Clients, and Locations

VF1  VF2  VF3

S1    S2    S3

C1    C2    C3
Mappings: Non-Optimal

Non-Optimal ⇒ More Delays

VFs

Washington University in St. Louis
http://www.cse.wustl.edu/~jain/talks/vm_distp.htm
Mappings: Optimal

Optimal mappings
Lesser Delays
Less Load

C1 → H1
C2 → H2
C3 → H3

VF1, VF2, VF3
Integer Linear Programming Formulation

- $G = \{V, E\}$ → Network
- $E \subseteq V \times V$ → Edges or Links
- $\lambda$ → Total VF instances ($\lambda_{min} \geq \lambda \geq \lambda_{max}$)
- $C$ → Capacity Matrix
  - $(C^1_1 + C^2_1 + C^3_1)$: Capacity of node 1 (CPU, Storage, NW)
- $P$ → Transmission Delay Matrix
- $W$ → Traffic Matrix
- $D$ → Demand Matrix
  - $(D^1_1 + D^2_1 + D^3_1)$: Demand of VF 1 (CPU, Storage, NW)
- $A$ → Allocation Matrix
- $T$ → Instance Matrix (How many instances of VF i are installed at server $j = T_{ij}$)
Minimization Function

- Minimize total delay without exceeding the server capacity

\[
\sum_{i \in |V|} \sum_{j \in |V|} \frac{(w_i \times A_{ij})}{p_{ij}}
\]

- P is a Transmission Delay Matrix,
- W is a Traffic Matrix,
- A is an Allocation Matrix.
Response Time (10-100 Nodes)

![Response Time Graph]

- 50-Nodes
- 60-Nodes
- 70-Nodes
- 80-Nodes
- 90-Nodes
- 100-Nodes

Total Response Time (Units)

VF Instances Installed
In the Cloud, a single ASP may have a set of services to be deployed.
Each server may host a specific set of VFs.
We propose a scheme for Dynamic Allocation of VFs to the servers.
Simple Example

- 5 VFs and 3 servers. Each with 2 resources (2-dimension)

<table>
<thead>
<tr>
<th>VFs</th>
<th>V1</th>
<th>V2</th>
<th>V3</th>
<th>V4</th>
<th>V5</th>
</tr>
</thead>
<tbody>
<tr>
<td>r₁</td>
<td>10</td>
<td>80</td>
<td>10</td>
<td>10</td>
<td>40</td>
</tr>
<tr>
<td>r₂</td>
<td>10</td>
<td>50</td>
<td>10</td>
<td>10</td>
<td>90</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Servers</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
</tr>
</thead>
<tbody>
<tr>
<td>r₁</td>
<td>50</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>r₂</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>
Demands

V1
V3
V4

V2

V5

Demands
Max-Min Allocation

- **S1**: 50 units
- **V1**, **V3**, **V4**: Max-Min Allocation
- **S2**: 100 units
- **V2**: Max-Min Allocation
- **S3**: 100 units
- **V5**: Max-Min Allocation
Minimum Residue Allocation

- Place a VF in a server where the remaining resources (residue) is minimized.
Average Utilization

![Average Utilization Chart]

- **VM1**: Max-Min: 40, Min-Residue: 50
- **VM2**: Max-Min: 60, Min-Residue: 80
Average Utilization of the VMs

Heuristic Comparison

- Minimum-Residue
- Max-Min

Maximum no of VMs vs Total Service Requests
Summary

1. We propose an Optimization model to *Locate* the Virtual Functions in the given network and *Allocate* the clients accordingly.

2. We propose a Heuristic scheme “*Minimum Residue*” for dynamic VF deployment.

3. The proposed scheme is compared against the standard *Max-Min* approach and improvements are showcased.