China Mobile: Yan Yang
Intel: Haibin Huang
CMCC Lab Overview
### Physical Lab Overview @CMCC

#### DC 1 (POD1)
- **Titanium Cloud 16.10 (Mitaka)**
- Controller nodes: 2, Compute nodes: 9

#### DC 2 (POD2)
- **VIO 4.0 (Ocata)**
- Controller node: 2
- Compute nodes: 6
- Hardware: RS300/E9000, 3 rack servers and 1 blade server with 6 blades

#### ONAP Deployment Env (POD3)
- **Titanium Cloud R4 (Newton)**
- Controller nodes: 2, Compute nodes: 3

#### ONAP Deployment Env & VNF Env (POD4)
- **Titanium Cloud 16.10 (Mitaka)**
- Controller nodes: 2, Compute nodes: 2

#### ONAP Deployment Env & VNF Env (POD5)
- **Titanium Cloud 16.10 (Mitaka)**
- Controller nodes: 2, Compute nodes: 3

- LOM Management, Admin, Private, Public networks inside
Use Case Deployment @CMCC

CMCC Domain: Casablanca

- **vCPE**
- **CCVPN**
- **vDHCP**
- **vAAA**
- **vBNG**
- **vGMUX**
- vxxx

VDF Domain: Casablanca

- **vCPE**

Integrate with 3rd Part Specific Components

POD3

- vCPE
- OS

POD4

- CMCC Domain: Casablanca
- vCPE
- OS
- SD-WAN Controller (Huawei)
- OTN Controller (Huawei)

POD5

- CMCC Domain: Casablanca
- vCPE
- OS
- SD-WAN Controller (ZTE)
- OTN Controller (ZTE)

POD1

- vBNG
- vGMUX
- vxxx

POD4

- (POD4)

POD5

- (POD5)
CMCC Lab key data

- ONAPs : 3
- Cloud : 5 (4) and vmware (1)
- Physical hardware : 73
- Participating member : 8
CCVNP Use Case
CCVPN Overview

CCVPN (Cross Domain and Cross Layer VPN) by ONAP Peering Orchestration Between SPs

- ONAP (CMCC)
  - SOTN Controller
  - SD-WAN Controller

- ONAP (VDF)
  - SOTN Controller
  - SD-WAN Controller

Diagram showing connections between sites and controllers:
- Site → Connection (E-Line) → UNI → Site
- Site → Connection (E-Line) → UNI → Site
CCVPN Network Topology

CMCC portal

CMCC ONAP

- SD-WAN Controller (HW)
- OTN Controller Server A (HW)
- OTN Controller Server B (HW)

Lab

- GE/10GE/OTU (HW)

Vodafone portal

Vodafone ONAP

- SD-WAN Controller Server (ZTE)
- OTN Controller Server (ZTE)

- uCPE (HW)
- vCPE (HW)

CMCC portal

- uCPE (HW)
- vCPE (HW)

Vodafone portal

- SD-WAN Controller Server (ZTE)
- OTN Controller Server (ZTE)

- uCPE (ZTE)
- vCPE (ZTE)

- OTN network (ZTE)

L3

L2

Lab

GE/10GE/OTU

OTN Domain A (HW)

OTN Domain B (HW)

OTN Domain A (HW)

OTN Domain B (HW)
CCVPN Test Case

- Topology Discovery
- Service Provisioning
- Closed Loop

- L2 Underlay (EoODU)
- L3 VNF
- Handoff

- OTN Domain 1
- OTN Domain 2

- ONAP(CMCC)

- ONAP(VDF)

- L2 Underlay (EoODU)
- L3 VNF
- Handoff

- SD-WAN Controller
- vGW
- WIND

- OTN Controller
- CPE
- L3 PNF

- CCVPN Service
vCPE Use Case
Key Test Case: ESR Registration, Model Design, Service Deployment, Service Recovery
vcPE

Design Time

SDC

UUI

Policy

VFC

OOF

AAI

Multi-cloud

Register as client for SDC

Distribute CSAR files

Translate VNF requirement into constraints needed by OOF

Search CSAR files

Return All distributed file

Trigger NS/VNF Package onboarding

Onboarding success/failure

Instantiate NS(Network Service)

Send out homing request

Retrieve homing policies for specific VNFs and services

Return homing policies

Get info of all available cloud regions including HPA

Return cloud regions info

Calculating the best solution

Return homing recommendation

Create Network/Port/Subnet/VM

Create completed

Instance NS(Network Service) success/failure

Register cloud region info
Design Time
vCPE – Package Design & Distribution

GW\BNG\... → SDC
- CPE E2E → 3. Service Package Design
- CPE NS → 2. NS Package Design
- 1. VNF Package Upload

SDC → Catalog
- Service Package → 4.2 E2E Service Pkg Distribution
- NS Package → 4.1 E2E Service Pkg Distribution
- VNF Package → 4.4 Query VNF Pkg

Catalog → Usecase UI
- 4.3 Query VNF Pkg

SDC → SO
- vCPE E2E Service Package

SDC → Policy
- vCPE E2E Service Package

SDC → VFC
- Policy
- vCPE E2E Service Package
- NS/VNFs Package

Usecase UI
- Usecase UI
vCPE – Package Design & Distribution

NS Design in SDC

VNF Package Onboarding in UUI

NS Package Onboarding in UUI
ESR Registration
VIM Configuration

- NIC configure
  We can refer to [1] before “Launching instances with SR-IOV ports”
  
  ```
  pci_passthrough_whitelist = [ 
  { 
    "devname": "eth3", 
    "physical_network": "physnet2"
  }]
  ```

- Create aggregate
  
  ```
  $ openstack aggregate create --property sriov_nic=sriov-nic-intel-1234-5678-physnet1:1 aggr11
  ```

- Create flavor
  
  ```
  $ openstack flavor create onap.flavor1 --id auto --ram 512 --disk 40 --vcpus 4
  $ openstack flavor set onap.flavor1 --property sriov_nic=sriov-nic-intel-1234-5678-physnet1:1
  ```

- Create provider network
MultiCloud register and HPA discovery

User adds Cloud-region

User adds Cloud-region to the inventory

Success/failure

Notify addition of new cloud-region

Selects the plugin based on VIM-type of that cloud-region

Trigger new cloud-region addition

Get flavors/VM Sizes/Instance types

Response with all types supported

Get details of flavors/VM Sizes/Instance Types learnt in previous step

Response (with details)

Normalize flavors/VM Sizes/Instance Types and HPA information into HPA-flavors as expected by ONAP

Add HPA-flavors and associated HPA-features and its attributes under the cloud-region
VIM & GVNFM Registration via ESR GUI

VIM Registration

Cloud Owner: CPE-DC
Cloud Region Id: RegionOne
Cloud Type: openstack
Cloud Region Version: titanium_cloud
Cloud Zone: z1
Physical Location Id: dl1
Username: vCPE
Password: ****
Auth Url: http://172.30.1.25000/v3
sslCACert: false
sslInsecure: true
Cloud Domain: Default
Default Tenant: vEPG

GVNFM Registration

Modify

Name: GVNFM DRIVER
Type: gvnfm driver
Vendor: vfc
Version: v1.0
URL: http://10.43.45.18:80/
VIM: CPE-DC RegionOne
Certificate URL: 
Username: 
Password: 
AAI information

It is expected that Openstack administrator creates alias that stars with sriov and put the vendor ID, device ID. Example:

Assume that there are two SRIOV-NIC cards supported by a region, Intel and Mellanox.

sriov-nic-intel-1234-5678-physnet1:1
sriov-nic-mellanox-2345-6543-physnet1:1
Run Time
ONAP – Run Time

Lifecycle: Instantiate/terminate/heal…

1. UUI
2. VF-C
3. OOF
4. Multi-Cloud
5. WindRiver (Openstack)
“flavorLabel": "vcpe.vgw",
"flavorProperties": [{
  "hpa-feature": "sriovNICNetwork",
  "mandatory": "True",
  "architecture": "generic",
  "hpa-version": "v1",
  "directives": [
    {
      "type": "sriovNICNetwork_directives",
      "attributes": [
        {
          "attribute_name": "vgw_nic_type", "attribute_value": "direct"},
        {
          "attribute_name": "vgw_provider_network", "attribute_value": "physnet1"
        }
      ]
    }
  ],
  "hpa-feature-attributes": [
    "hpa-attribute-key": "pciVendorId",
    "hpa-attribute-value": "1234", "operator": ",", "unit": ""
  },
    "hpa-attribute-key": "pciDeviceId",
    "hpa-attribute-value": "5678", "operator": ",", "unit": ""
  },
    "hpa-attribute-key": "pciCount",
    "hpa-attribute-value": "1", "operator": ",", "unit": ""
  },
    "hpa-attribute-key": "physicalNetwork",
    "hpa-attribute-value": "physnet1", "operator": ",", "unit": ""
  ]
},
]
VF-C — OOF Homing

1. VFC sends out homing request to OOF(OSDF) containing resource info
2. OOF(OSDF) pulls all the related homing constraints from Policy
3. OOF(HAS) check AAI database to pull region(flavor) information
4. OOF(OSDF) returns homing allocation solution to VFC
OOF Homing

hpa-capability-id="ty53fd3d-0b15-11w4-81b2-6210ef6dff9",
hpa-feature="sriovNICNetwork",
architecture="intel64",
hpa-version="v1",

<table>
<thead>
<tr>
<th>hpa-attribute-key</th>
<th>hpa-attribute-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>pciCount</td>
<td>{value: 1}</td>
</tr>
<tr>
<td>pciVendorId</td>
<td>{value: &quot;1234&quot;}</td>
</tr>
<tr>
<td>pciDeviceId</td>
<td>{value: &quot;5678&quot;}</td>
</tr>
<tr>
<td>physicalNetwork</td>
<td>{value: &quot;physnet1&quot;}</td>
</tr>
</tbody>
</table>

"flavorLabel":"vcpe.vgw",
"flavorProperties":[{
  "hpa-feature": "sriovNICNetwork",
  "mandatory": "True",
  "architecture": "generic",
  "hpa-version": "v1",
  "directives": [
    {
      "type": "sriovNICNetwork_directives",
      "attributes": [
        {"attribute_name": "vgw_vnic_type", "attribute_value": "direct"},
        {"attribute_name": "vgw_provider_network", "attribute_value": "physnet1"}
      ]
    }
  ],
  "hpa-feature-attributes": [
    {"hpa-attribute-key": "pciVendorId",
      "hpa-attribute-value": "1234",
      "operator": ":",
      "unit": ""
    },
    {"hpa-attribute-key": "pciDeviceId",
      "hpa-attribute-value": "5678",
      "operator": ":",
      "unit": ""
    },
    {"hpa-attribute-key": "pciCount",
      "hpa-attribute-value": "1",
      "operator": ">=",
      "unit": ""
    },
    {"hpa-attribute-key": "physicalNetwork",
      "hpa-attribute-value": "physnet1",
      "operator": ":",
      "unit": ""
    }
  ]
}]

AAI

Policy
VFC Create network and port

```
"type": "sriovNICNetwork_directives",
"attributes": [
    {"attribute_name": "vgw_vnic_type", "attribute_value": "direct"},
    {"attribute_name": "vgw_provider_network", "attribute_value": "physnet1"}
]

a. Create network, ports
   # openstack network create --provider-network-type vlan --provider-physical-network physnet1 tst-network
   # openstack port create --name sriov-port --binding:vnic-type direct

b. Create server
   # openstack server create --flavor onap.hpa.flavor11 --image Ubuntu_16.04 --nic port-id= test-sriov-nic
```
SO and VFC will call Framework
Thank you!