OSM-MR#9 Hackfest
Placement optimization for our Network Services
Lars-Göran Magnusson (Arctos Labs)
What do we mean by Placement Optimization?

- Placement in context of OSM is the process of deciding **which VNF goes into which VIM**

- Optimal is subject to:
  - Cost of compute in VIMs
  - Cost of links for NS interworking
  - Constraints in NS interworking (Latency, Jitter) – if there are any

- Placement feature makes this process **Automatic & Optimal**

---

**Business Service Basic Architecture**, from *OSM Deployment and Integration WP, Feb 2020*
The Optimization Process

• Placement function
  • Will consider all VIM’s available to the user
  • Will make sure constraints are met – if there are any
  • Will optimize Cost (the Criteria)

• I.e. select the option that fulfills constraints at the lowest possible cost
  • Modeled as a constraints optimization problem

Computation of optimal placement of VNFs over VIMs by matching NS specific requirements to infrastructure availability and run-time metrics, while considering cost of compute/network.
Placement optimization examples

1. **Cost** optimization only

Constraint: Nothing
Criteria: Cost

2. **Cost** optimization with Latency constraint

Constraint: Latency / Jitter
Criteria: Cost

3. Utilization optimization with Latency constraint

Constraint: Latency
Criteria: Utilization

4. **Cost** optimization with Capability constraint

Constraint: Capability (e.g. CPU / GPU etc)
Criteria: Cost
Examples of use cases

**UPF supporting Low-latency**
Placement of UPF close to customer to achieve latency constraint

**Transport optimization (cost) for Application components**
Placement of Application components close to the source of data to reduce transport cost/load

**Compute cost optimization for slicing**
Placement of CN VNF’s at most cost effective compute

Deploy as close as it has to be

Deploy as far away as it can be
Example 1:
VIM#2
Auto
Auto
Auto
VIM#3

Example 2:
VIM#2
Auto
Auto
Auto

Example 3:
Auto
Auto
Auto
Auto

• Ability to “pin” a VNF to e.g.
  • the VIM with a specific VNF (e.g. P-GW)
  • the VIM with connectivity to a PNF
  • a CPE (customer location)

*Auto implies there is no VIM specified, this placement is therefore subject to placement optimization*

=> this is what Placement is all about – finding out where VNFs should (or must) be deployed in a multi-VIM NFVI
Some different scenarios

Scenario a:
- VIM #3
- Pinning: Auto
- Constraints: Auto

Scenario b:
- VIM #3
- Pinning: VIM #3
- Constraints: Auto

Scenario c:
- VIM #4
- Pinning: Auto
- Constraints: latency=15

Topologies & Cost:
- Cost: 10
- Global DC VIM #1
  - Cost: 30
  - Latency: 30
  - Cost: 5
  - Latency: 20
- Local DC VIM #2
  - Cost: 20
- VNF #1
  - vld_one_vld
- VNF #2
  - vld_two_vld
- VNF #3

© ETSI 2020
Install and configure PLA in OSM
The PLA component in OSM

- Basic functionality initially
- Automatic placement is optional, invoked by the user at instantiate of Network Service
  - `--config '{placement-engine: PLA, placement-constraints: {}, ...}'`
  - Constraints given in the instantiation request
  - Will consider placement over the VIMs available to the user
- Interacts with LCM, Common Services

- New component
  - Optional, install with `--pla`
Configure PLA

- You need two configuration files
  - vnf_price_list.yaml
  - pil_price_list.yaml

- The configuration files are copied to the PLA container using the following commands:
  $ docker cp vnf_price_list.yaml $(docker ps -qf name=osm_pla):/placement/.
  $ docker cp pil_price_list.yaml $(docker ps -qf name=osm_pla):/placement/.

The price list for compute determines the price for each VNF at each VIM. The file (vnf_price_list.yaml) is written in Yaml.

The price list and characteristics for transport links between VIMs (PoP Interconnecting Link – PiL). In current release the price is given per link without any consideration to BW or other QoS parameter. The file (pil_price_list.yaml) is written in Yaml.

Note: In current OSM release the link characteristics are hard coded into this file, in future releases this data should be retrieved from the infrastructure by monitoring mechanisms.

© ETSI 2020

Note: Don’t copy the files as hackfest participant, it’s already done!
Invoke PLA

1. Request Placement Cost Optimization
   --config '{placement-engine: PLA}'

2. Request Placement Cost Optimization with pinning of specified VNF
   --config '{placement-engine: PLA, vnf: [{member-vnf-index: "1", vim_account: OpenStack3}]}'

3. Request Placement Cost Optimization with VLD Constraints
   --config '{placement-engine: PLA, placement-constraints: {vld-constraints: [{id: vld_1, link-constraints: {latency: 120, jitter: 20}}, {id: vld_2, link-constraints: {jitter: 20}}]}}'

4. Combo of 2 and 3

Note: GUI is also supported, with or without YAML file
Hands-on: Placement of the Magma AGW + emulator VNF
Hands-on cancelled

- We have to cancel the hands-on session
- To do the hands-on we need a second VIM
- Unfortunately our second VIM is at the moment not available in the MR#9 hackfest environment

- What now?
  - Walkthrough of the hands-on
  - Demo
Objective: create a new slice, sharing the Magma orchestrator, automatically deployed to another DC
Launch a 2\textsuperscript{nd} slice

- Create another VIM
  - The vim name is important, it must match content of the vnf_price_list.yaml file
  - `osm vim-create --name etsi-openstack-lowcost --user osm_hackfest --password osm_hackfest --auth_url http://172.21.7.5:5000/v3 --tenant osm_hackfest --account_type openstack --config '{management_network_name: osm-ext, dataplane_physical_net: physnet2, microversion: 2.32}'`
  - --user, --password and --tenant follows your personal settings for the hackfest
  - Don't forget the additional configuration

- List the vims, and collect the new vim uuid, we need it in the next step
  - `osm vim-list`

Cancelled
Launch a 2\textsuperscript{nd} slice

- Register the PDU to the 2\textsuperscript{nd} vim account

1) Edit pdu.yaml

```
name: router01
description: router
type: gateway
vim_accounts: [94c1218a-e9c7-42d8-b0ae-6de0d0a635ae]
shared: false

interfaces:
  - name: eth0
    ip-address: 172.21.250.200
    mgmt: true
  - name: eth1
    ip-address: 192.168.239.7
    mgmt: false
```

Enter the uuid for the new vim to the vim_accounts list

2) Launch the pdu-create command

```
$ osm pdu-create --descriptor_file pdu.yaml
```

- Note: You may also use the GUI (Instances -> PDU Instances) to register the PDU

© ETSI 2020
Launch a 2\textsuperscript{nd} slice

- Prepare for PLA support – modify the configuration file
  - make a copy of params_slices.yaml

```yaml
netslice-subnet:
  - id: slice_hackfest_nsd_epc
    placement-engine: PLA
    wimAccountId: False
    additionalParamsForVnf:
      - member-vnf-index: '1'
        additionalParams:
          agw_id: 'agw_101'
          agw_name: 'AGW101'
          orch_ip: '172.21.251.XXX'  ## change this to the MetalLB IP address of your orc8r_proxy service.
          orch_net: 'osmnet'

  - id: slice_hackfest_nsd_epcmgmt
    additionalParamsForVnf:
      - member-vnf-index: 'orc8r'
    additionalParamsForKdu:
      - kdu_name: orc8r
        additionalParams:
          proxyserviceloadBalancerIP: '172.21.251.XXX' # MetalLB IP Address
```

Uncomment placement-engine: PLA
Uncomment wimAccountId: False

Need another agw_id, agw_name e.g. 101

- Create the slice

```
osm nsi-create --nsi name magma slice 2 --nst name magma slice hackfest nst \--config_file params_slices2.yaml --ssh_keys ~/.ssh/id_rsa.pub --vim_account etsi-openstack-x
```

Use your ordinary vim\_account
Launch a 2\textsuperscript{nd} slice

- Check where the vnf ended up
  
  \texttt{osm vnf-list}

- \texttt{vim\_account\_id} should correspond to \texttt{etsi-openstack-x-lowcost} for the new slice

- same Magma \texttt{orc8r} as before

- You may configure and send traffic over the new slice

- Clean up: delete the slice
  
  \texttt{osm nsi-delete <nsi\_name> or <nsi\_id>}

© ETSI 2020