# K-ONE 기술 문서 #37

# Optimization models for placing high availability requests over cloud infra

Document No. K-ONE #37
Version 0.2
Date 2018-12-27
Author(s) Truong-Xuan Do

K-ONE

ONK
OpenNetworkingKorea

■ 문서의 연혁

| 버전 | 날짜 | 작성자 | 내용 |
|---|---|---|---|
| 0.1 | 2018.11.30 | Truong-Xuan Do | Draft |
| 0.2 | 2018.12.26 | Truong-Xuan Do | Profreeding |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

본 문서는 2015년도 정부(미래창조과학부)의 재원으로 정보통신 기술진흥센터의 지원을 받아 수행된 연구임 (No. B0190-15-2012, 글로벌 SDN/NFV 공개소프트웨어 핵심 모듈/기능 개발)

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (No. B0190-15-2012, Global SDN/NFV OpenSource Software Core Module/Function Development)

# 기술문서 요약

네트워크 기능 제공시 안정적으로 네트워크 서비스를 제공하려면 고가용성이 보장되어야 한다. 가상환경에서 제공되는 네트워크 기능의 가용성 수준을 보장하기 위해서는 서로 다른 중복 모델 또는 클러스터 기능이 하나의 솔루션으로 사용될 수 있다. 따라서 본 기술 문서에서는 지리적으로 분산 된 클라우드 인프라 스트럭처에 서로 다른 중복 모델을 사용하여 클러스터 요청을 배치하는 최적화 모델을 설계하고자 한다. 이 모델에서는 다양한 리소스 요구사항을 고려하였으며 현재 배치 구조를 고려한 알고리즘을 제안하였다. 또한 제시된 알고리즘을 프로토타입으로 구성하였다.

# Contents

## K-ONE #1. K-ONE 기술문서 템플릿

# 그림 목차

# 표 목차

## 1. High availability in NFV

To increase the availability level of the network services provided by the VNFs, in the NFV environment, the VNFs can be deployed with different redundancy models [1]-[5], such as active-standby or active-active, as shown in Fig. 1 and Fig. 2. Fig. 1 depicts the active-standby configuration. In the NFV environment, a virtual router is used to map between external connection point (connected to external network) to internal connection point of each VNF. In the case of active VNF failure, the MANO should reconfigure the virtual router to map the external connection point to the internal connection point of standby VNF. In addition, to keep the system's state information consistent, the state replication is required. The active VNF stores state information of ongoing sessions to the external storage and replicates the state to the storage of the standby VNF. Fig. 2 depicts the active-active configuration. In this configuration, two external connection points are configured to map to two internal connection points, respectively. The client can use services provided by one of active VNFs by selecting the IP address of one of external connection points. The selection can be supported by the DNS service or a load balancer. The detailed analysis of each high availability scenario in the NFV environment including failure detection and recovery process is presented in [5]. In the NFV environment, the VNFs are instantiated and configured from a central MANO across geographically distributed clouds. Therefore, the redundancy models for the VNFs can be easily deployed and configured across the geo-distributed cloud infrastructures. This enhances the availability and reliability of virtual network services in the cases of natural disasters, which affect a large geographical area.

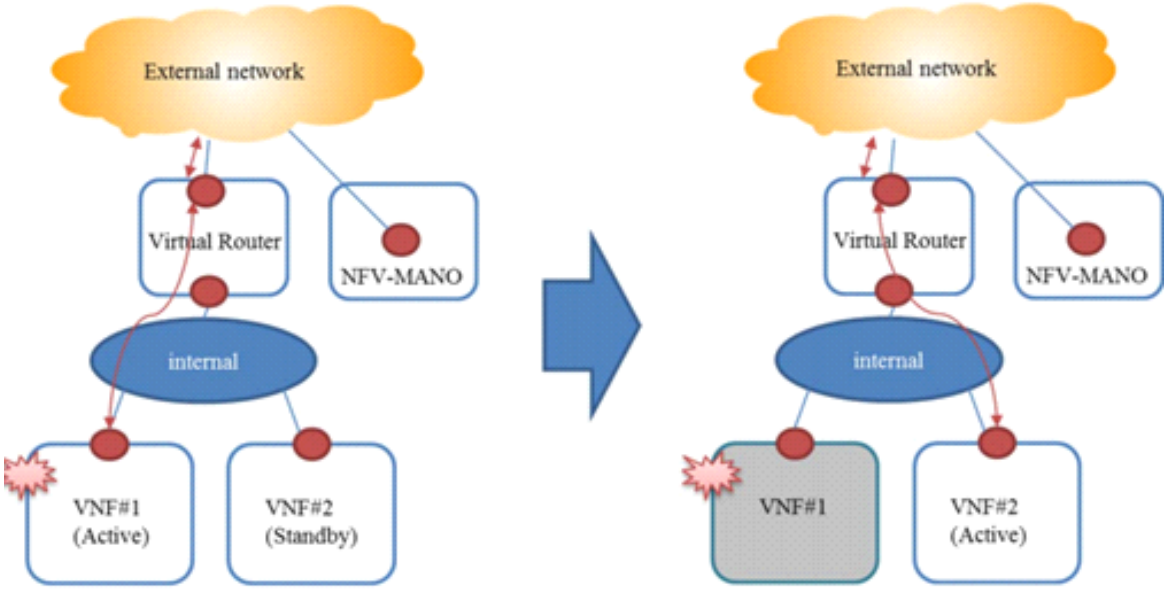


Figure 1. Active-Standby model

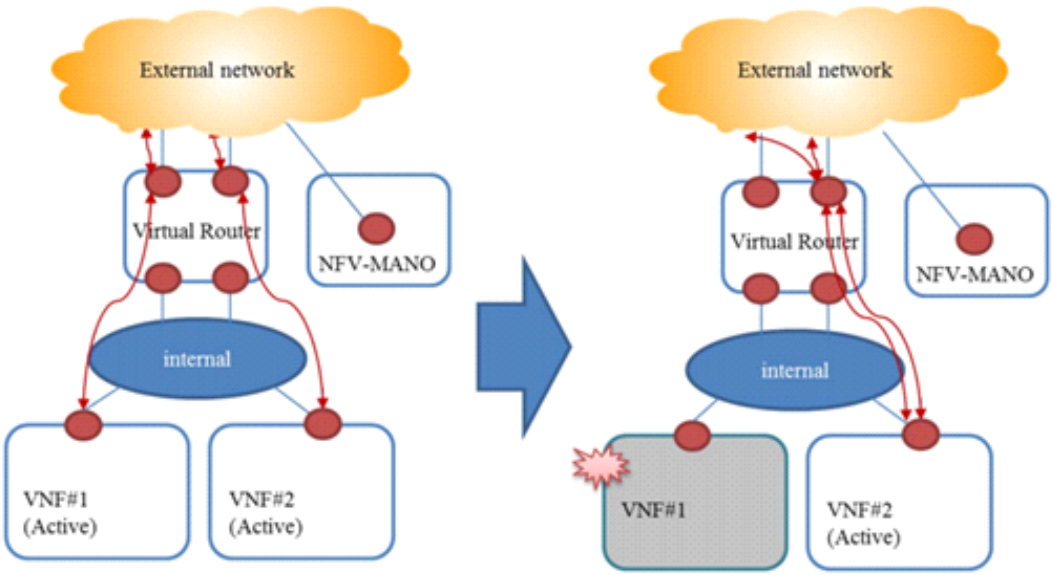


Figure 2. Active-Active model

## 2. Deployment architecture

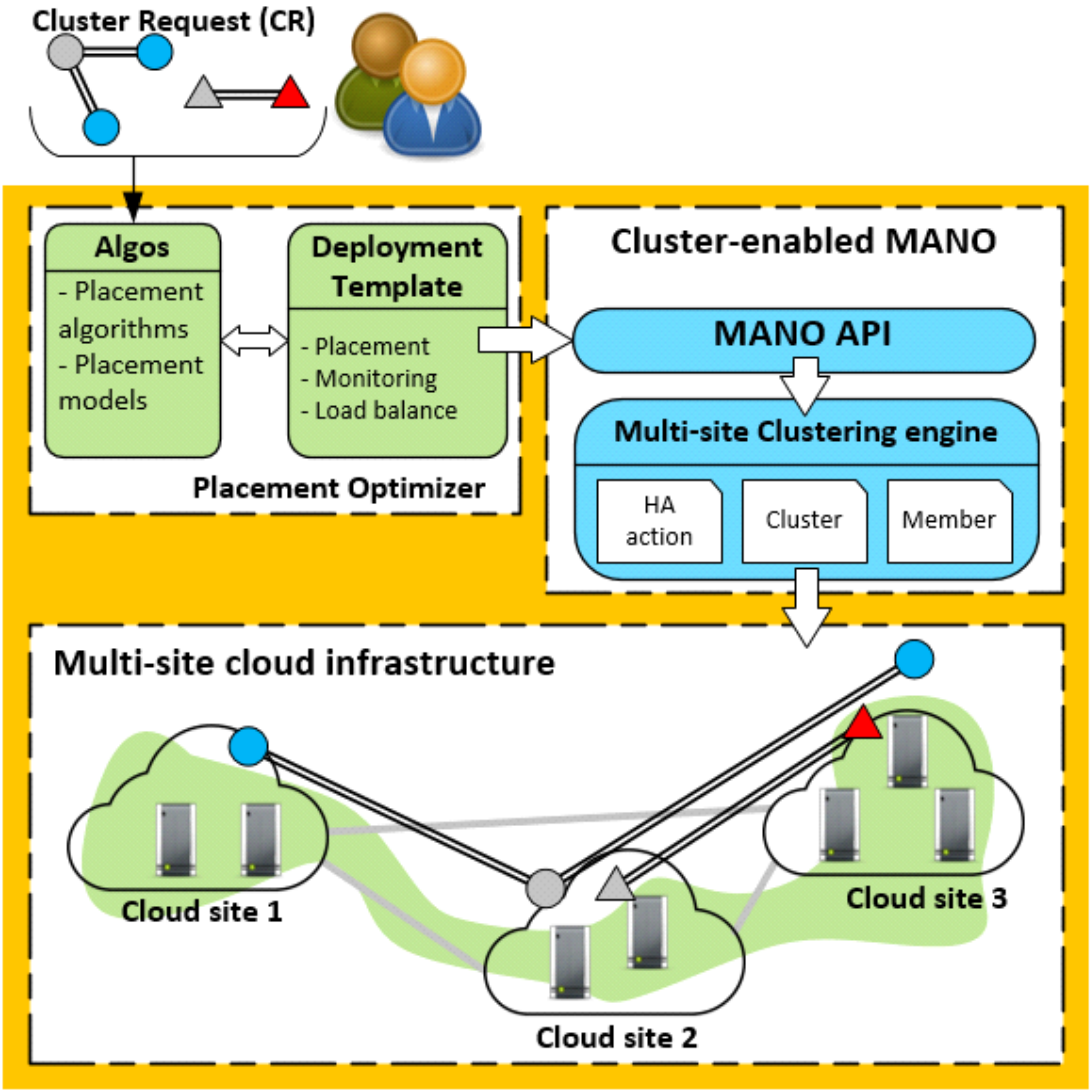


Figure 3. Deployment architecture

In order to deploy cluster requests (redundancy models) over geo-distributed cloud infrastructure, we need to develop placement models and placement algorithms. When we run this models and algorithms, the output will be written in a template file. The template includes the description on the deployment location of each VNFs, monitoring policies when failure, and load balancing configuration. A cluster-enabled MANO is an management and orchestration framework, which has the clustering management function embedded. The MANO has two parts. The first part is MANO APIs that exposes the APIs for creating and managing cluster requests. The second part is cluster engine that

includes abstract classes representing required objects for cluster management, such as cluster, members in the cluster, and actions. Multi-site cloud infrastructure includes different cloud centers, which are distributed over different geographical locations.

### 3. Resource demand calculation

We assume that all general cluster requests are decomposed into three primitive forms: one active and multiple standby, multiple active and one standby, and multiple active, as shown in Fig. 4.

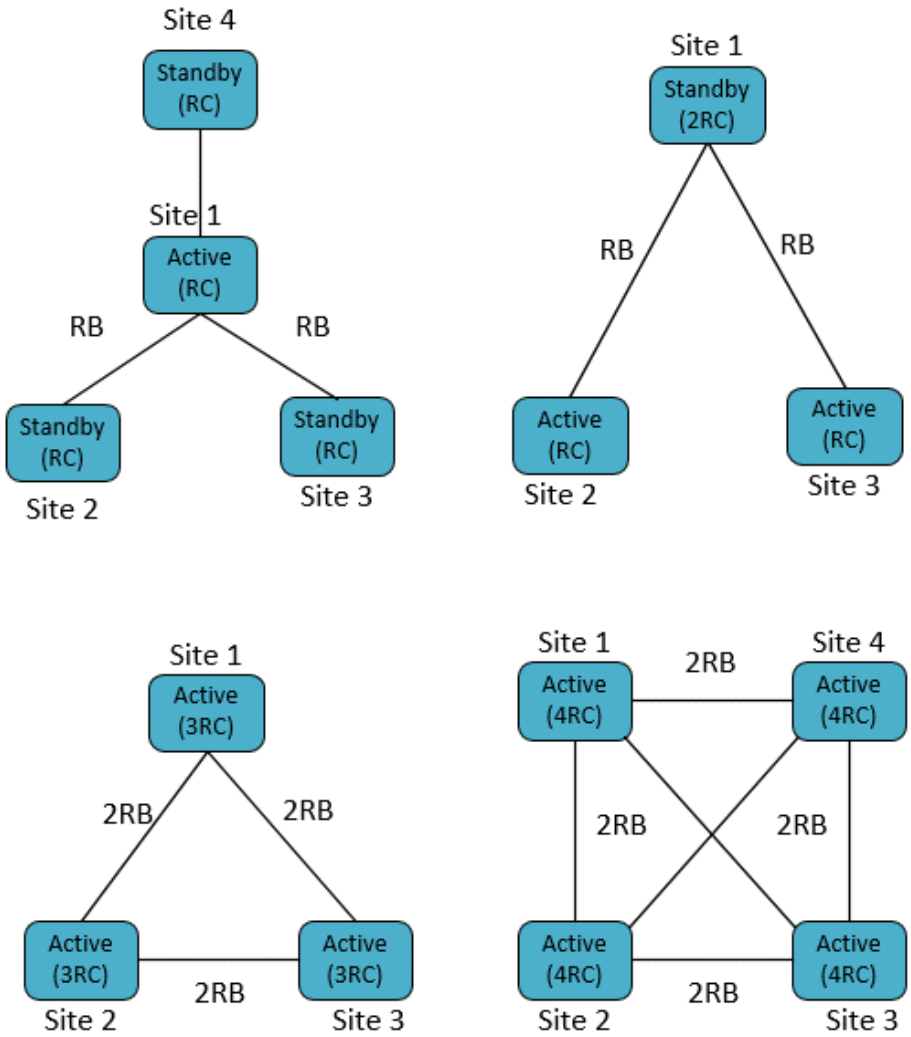


Figure 4. Redundancy models

The resource demands for VNFs and virtual links are calculated as follows. RC

denotes the computing resource demand for active and standby VNFs. RB denotes the bandwidth resource demand for virtual links between VNFs.

*One Active multiple Standby:*

$$RC_a = RC$$
$$RC_s = RC$$
$$RB_l = RB$$

*Multiple Active one Standby:*

$$RC_a = RC$$
$$RC_s = \sum_{a \in ACT_r} RC$$
$$RB_l = RB$$

*Multiple Active only:*

$$RC_a = \sum^{a \in ACT_r} RC$$
$$RB_l = 2RB$$

## 4. Placement models for different high availability clusters

We devise an optimization model to place VNFs and map virtual links of different availability cluster requests over geo-distributed cloud infrastructure. We denote a set of cloud center by N and a set of physical links connecting cloud centers by E. Each cloud center has limited amount of computing resources and each virtual link has limited amount of bandwidth resources. Each cloud center and physical link have a value of availability. We define our requests which have the form of three redundancy models discussed above. The math notation is presented in the Fig. 5.

Our problem is defined as follows:

For each cluster request, we try to solve two problems. The first problem is placement problem that attempts to place VNFs in each cluster request on cloud centers. The second problem is routing problem that attempts to map virtual links in each cluster request on physical links. The optimization goal is to minimize the network bandwidth resource consumption while satisfying the availability requirement. Three constraints are considered: computing resource constraints, physical network bandwidth constraint, and minimum availability

constraints.

| Notation | Description |
|---|---|
| $N$ | Set of cloud centers |
| $E$ | Set of WAN links between cloud centers |
| $CR$ | Set of cluster requests |
| $ACT_r$ | Set of active VNFs in cluster request $r$ |
| $STB_r$ | Set of standby VNFs in cluster request $r$ |
| $NU_r$ | The total number of VNFs in cluster request $r$ |
| $VL_r$ | Set of virtual links between VNFs in request $r$ |
| $RC_a$ | Computing resource demand for active VNF $a$ |
| $RC_s$ | Computing resource demand for standby VNF $s$ |
| $RB_l$ | Bandwidth resource demand for virtual link $l$ |
| $CA_n$ | Computing resource capacity of cloud center $n$ |
| $BW_e$ | Bandwidth resource capacity of physical link $e$ |
| $A_n$ | Availability value of cloud center $n$ |
| $A_e$ | Availability value of physical link $e$ |
| $A_r$ | Availability of cluster request deployment $r$ |
| $AL_s$ | Availability of virtual link to standby function $s$ |
| $BC_e$ | Bandwidth consumption of physical link $e$ |
| $CC_n$ | Computing consumption of cloud center $n$ |

Figure 5. Math notations

Three decision variables are used: X, Y, and U. X, Y equal 1 if active function and standby function in the cluster request are placed on cloud centers, and 0 otherwise. U equal 1 if virtual link in the cluster request is mapped on the physical link, and 0 otherwise. The optimization problem is present in mathematic form as follows:

**Objective:**

$$\textbf{Minimize} \sum_{e \in E} BC_e \tag{6}$$

*Computing resource constraint:*

$$\forall n \in N :$$

$$CC_n = \sum^{r \in CR} ( \sum_{a \in ACT_r} X_n^a RC_a + \sum_{s \in STB_r} Y_n^s RC_s) \leq CA_n \tag{7}$$

*Link bandwidth constraint:*

$$\forall e \in E :$$

$$BC_e = \sum_{l \in VL_r}^{r \in CR} U_e^l RB_l \leq BW_e \tag{8}$$

*Availability constraint:*

$$\forall r \in CR :$$

$$A_r \geq A_r^{min} \tag{9}$$

*Deployment policy constraint:*

$$\forall r \in CR, n \in N :$$

$$\sum_{a \in ACT_r} X_n^a + \sum_{s \in STB_r} Y_n^s < NU_r \tag{10}$$

Figure 6. Optimization problem

## 5. Proposed solutions

### 5.1. Greedy solutions

Key concepts of greedy solutions is to place VNFs according to one of following policies:

+ Best bandwidth resources: cloud centers with the highest bandwidth resource will be selected first

+ Highest availability: cloud centers with the highest availability values are selected first.

+ Best computing resources: cloud center with highest computing resources are selected first.

**Algorithm 1** Greedy algorithms

1: **Input**: $N, E, CR, A_n, A_e, CA_n, BW_e$
2: **Begin**:
3: $RC_a, RC_s, RB_l \leftarrow calResourceDemand()$
4: Sort $CR$ by $requestSize$ in a decreasing order
5: Sort $N$ according to chosen greedy policies
6: **for** $rinCR$ **do**
7: $X, Y, U =$ **RP-SR**$(r, RC, RB)$
8: **end for**
9: **Finish**
10: **Output**: $X, Y, U$

Figure 7. Greedy algorithms

1: **Input**: $X, Y, virLinks, RB$
2: **Begin**:
3: **for** $l \in virLinks$ **do**
4: Obtain $m, n$ if $X_m^{l[0]} = 1$ and $X_n^{l[1]} = 1$
5: Create a $Graph$ with vertices, edges $N, E$
6: Set cost of edge to 1 if $BW_e \geq RB_l$
7: Set cost of edge to 100 if $BW_e \leq RB_l$
8: $P(n, m) \leftarrow Graph.Dijkstra(n, m)$
9: **if** All links in $P(n, m)$ have cost 1 **then**
10: $l$ can be mapped on $P(n, m)$
11: Update $U$
12: **end if**
13: **end for**
14: **Finish**
15: **Output**: $U$

Figure 8. Shortest path routing

First, we calculate resource demand for each cluster request using the function calResourceDemand(). Then, we sort cluster requests by the size of request in a deceasing order. We in turn select each cloud center according to greedy policy to place request. Then, for each request, we call algorithm reliable placement and shortest path routing (RP-SR) to place and map each cluster request. The RP-SR firest selects the best cloud centers according to greedy policies to place VNFs. The RP-SR always check the computing resource constraints and availability constraints during placement. If the computing resource constraint is

not satisfied, the another center will be selected. If the availability constraint is not satisfied, the more distributed the cluster request is. After placement, the routing problem is solved using the shortest path routing algorithm or Dijiktra algorith, as shown in Fig. 8.

### 5.2. Topo-aware algorithm

Normally, the greedy algorithms are not optimal in terms of network resources. we attempt to devise a new algorithm that places cluster request more efficiently. As shown in Fig. 9, the algorithm attempts to place cluster requests according to their topology. First, we also calculate resource demand and sort cluster request by its size. Then, depending on the topology, we will place requests using different functions. For the star topology, we try to build a tree to place the request and for the mesh topology, we try to select the best combination that has the minimum number of usable links to place requests.

**Input**: $N, E, CR, A_n, A_e, CA_n, BW_e$
**Begin**:
$RC_a, RC_s, RB_l \leftarrow calResourceDemand()$
Sort $(CR, requestSize)$ in a decreasing order
**for** $r \in CR$ **do**
    Determine request as star or mesh topology
    **if** Star topology **then**
        $PlaceStarTopo(N, E, CR, A_n, A_e, CA_n, BW_e, RC, RB)$
    **end if**
    **if** Mesh topology **then**
        $PlaceMeshTopo(N, E, CR, A_n, A_e, CA_n, BW_e, RC, RB)$
    **end if**
**end for**
**Finish**
**Output**: $X, Y, U$

Figure 9. Topology-aware approach

## 6. Prototype results

Currently, there has been one research work dealing with clustering management for VNFs in OpenStack Tacker [6]. However, this proposal is just limited to one site cloud region, hence it could not meet the high level of

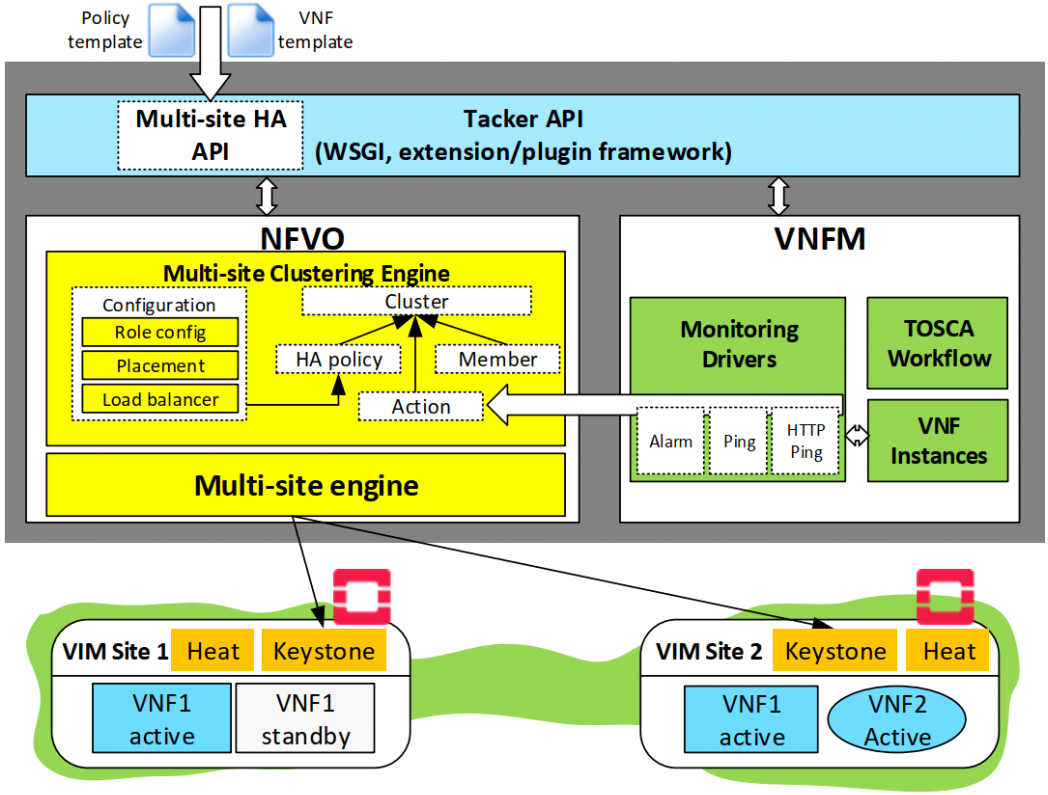


Figure 10. Tacker architecture to support high availability deployment

reliability for VNF over multiple clouds. Besides, OPNFV multisite [7] focuses on the enhancement of OpenStack to support multi-site NFVI clouds. This project considers how to provide application-level redundancy and network topology spanning multiple sites from a single point. However, the clustering management related features for multi-site are not implemented yet. Therefore, we would like to propose policy-based clustering service engine for multi-site NFVI cloud.

In our proposed architecture, OpenStack is used as a virtual infrastructure manager (VIM), where VNFs can be hosted. We consider Tacker as MANO project that manages and orchestrates cluster resources on top of VIMs from a single point. We extend Tacker architecture aiming at providing multi-site clustering service for VNFs across multiple OpenStack deployments. Fig. 10 shows proposed architecture that comprises of extended APIs, multi-site. Multi-site HA APIs are used to provide an interface for users to access multi-site clustering management functions of NFVO Tacker. The multi-site

clustering engine includes Cluster, Member, Action, and HA policy. A Member includes information about included VNFs, which is extracted from VNF descriptor (VNFD). A Cluster includes information about VNF identity, site location of VNF, and the assigned role of VNF (e.g. Active or Standby). The Cluster supports CRUD operations via HA API. And the Action includes recovery actions for a cluster when a failure occurs (e.g. change Standby to Active and deploy a new Standby). Recovery action is included in the VNFD as a triggered action that should be executed when there is a failure notification event triggered by the monitoring drivers. Furthermore, we also introduce a policy template that includes placement strategies from optimization step that will be used in deployment step.

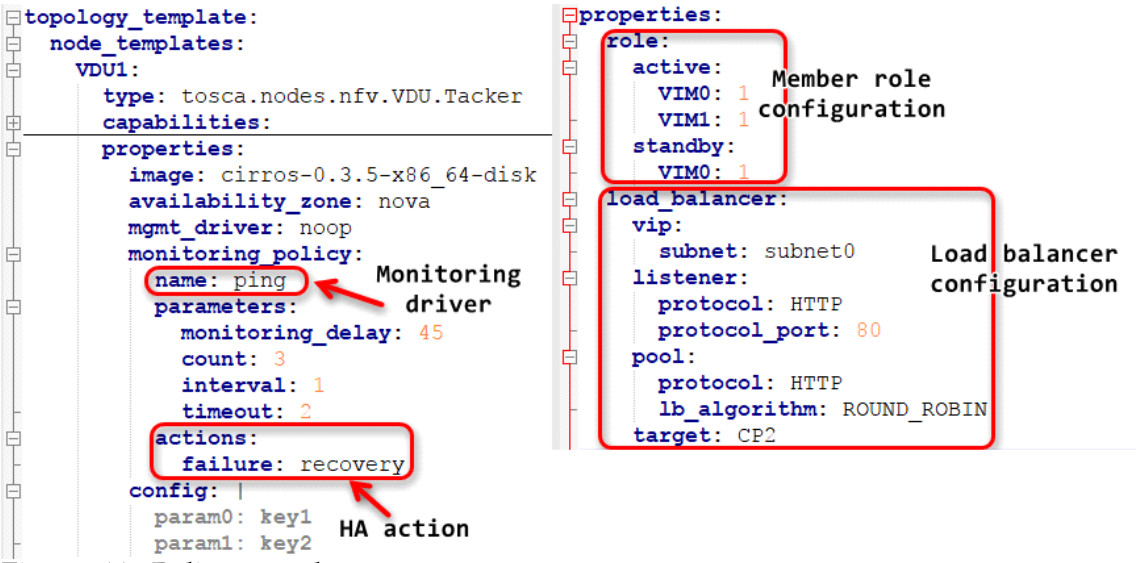

Figure 11. Policy template

The multi-site cluster management is based on predefined HA policy template, which consists of placement policy, role configuration, and load balancer configuration. The VNF placement policies are used to specify the deployment locations of member roles in a cluster, while the load balancer configuration defines parameters to configure traffic forwarding rules for steering traffic into VNFs in a cluster. To enable policy-based clustering management, right after the creation of a new multi-site cluster, the predefined HA policies will be attached to that cluster. The existing monitoring drivers of Tacker are also connected to Actions of multi-site cluster engine. The monitoring drivers used to monitor the status of the VNFs, including survival of VNFs (e.g. Ping driver) or working

load of VNFs (e.g. alarm-based monitoring). Depending on the type of configured monitoring driver, when the VNF dies or working load is over the threshold, an alarm will be delivered to the multi-site cluster management engine. The multi-site clustering engine can check the received alarm message and trigger the corresponding recovery actions based on predefined policies attached to the cluster.

We performed the experiments of cluster deployment in three configurations, with one, two and three cluster nodes. Fig. 12 show the average execution time. As the absolute values of the execution time vary widely based on environments, we chose to normalize all values for simple comparison. Here, we can see how the execution time increases with growth of the number of required nodes. It is observed that the execution time mainly increases when new nodes are required to deploy in the new cloud sites. The main reason for this gap is the time required for an additional load balancer creation on the new sites. On the other hand, the service down time of the clustering service for multiple sites is represented in Fig \ref{im-down-time}. As expected, the service down time does not depend on multiple sites configuration, since there is no required time for load balancer operation.

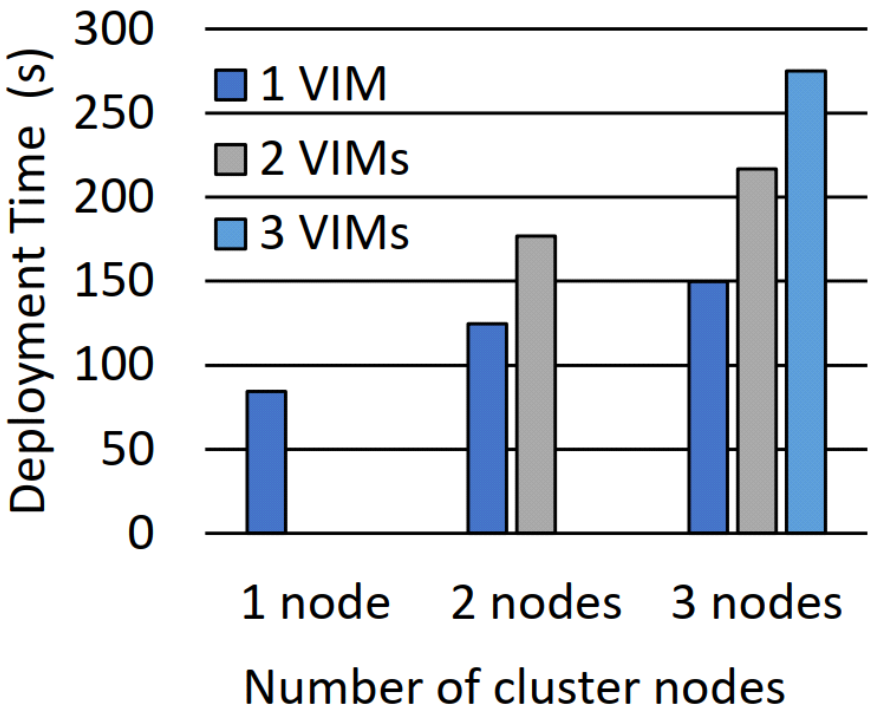


Figure 12. Experiment results

## References

[1] Vargas E., BluePrints S., “High availability fundamentals”. Sun Blueprints series, 2000.

[2] ETSI GS NFV-REL 001 v1.1.1, “Network Functions Virtualization (NFV); Resiliency Requirements”, 2015.

[3] ETSI GS NFV-REL 002 V1.1.1, “Network Functions Virtualization (NFV); Reliability; Report on Scalable Architectures for Reliability Management”, 2015.

[4] ETSI GS NFV-REL 003 V1.1.1, “Network Functions Virtualization (NFV); Reliability; Report on Models and Features for End-to-End Reliability”, 2016.

[5] OPNFV High availability, Scenario Analysis of High Availability

[6] Openstack Tacker, a generic VNF manager and NFVO

[7] OPNFV Multiste, Multi-site virtualized infrastructure

# *K-ONE 기술 문서*



작성기관: K-ONE Consortium
작성년월: 2018/12