Analytic Framework for NFV Orchestrator
draft-liu-nfvrsg-analytic-framework-00.txt

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 21, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents...
carefully, as they describe your rights and restrictions with respect to this document.

Abstract

This draft introduces an analytic model and framework with data collection and policy management in NFV Orchestrator.

Table of Contents

1. Introduction ................................................ 2
2. Data collection and policy inventory......................... 3
   2.1. Data collection framework................................ 3
   2.2. Policy inventory........................................ 4
3. Analytic model and framework................................ 5
   3.1. The real-time analytic mode ............................ 6
   3.2. The non-real-time analytic model........................ 8
4. Analytic framework in NFV Orchestrator..................... 9
5. Security Considerations.................................... 10
6. IANA Considerations ....................................... 10
7. Conclusions ................................................ 10
8. References ................................................ 11
   8.1. Normative References................................... 11
   8.2. Informative References................................ 11
9. Acknowledgments ............................................ 11

1. Introduction

As NFV being researched, we have observed more and more research started focusing on NFV Orchestrator. Most research in Orchestrator is to address issues around orchestrator architecture, task fulfillment, status monitoring as well as network analytic and policy management.

In this draft, we would like to focus on discussing the network analytic and policy management (FAPM) module. After that, we will introduce some Orchestrator architecture, feature and the relationship with FAPM module. We propose this draft because we believe FAPM will play an important role in future network orchestration. We have researched and implemented the features of orchestration task fulfillment and VIM resource monitoring. In the GUI, the network operator can define Network Service, manage VNF lifecycle (e.g. create, terminate, scale in/out), monitor its status from VIM ceilometer. However, the current network policy is simple and pre-set in orchestrator. If the FAPM module can be integrated into this orchestration loop (monitor->FAPM->fulfillment), we may
create a more ‘intelligent’ network orchestration and policy management.

In section 2 of this draft, we would like to introduce the capability of data monitor and collection attributes by Orchestrator. Besides, the policy inventory is also important to introduce in this section.

In section 3, we proposed the analytic model and framework for FAPM module. The real-time analytic model and the non-real-time analytic model are described respectively. We present one data mining techniques for the real-time analytic model, and provide the expectation for the non-real-time analytic model.

In section 4, we describe the Orchestrator architecture along with introduction of how FAPM module fits in this architecture.

2. Data collection and policy inventory

2.1. Data collection framework

The orchestrator is more like a central brain to provide a global manage and control of the network resource as well as the logical network service (e.g. NS model, SFC). By our research, we have onboard of several network elements (VNF) from different venders, such as vEPC, vIMS, vCPE, vBRAS and Nanocell Gateway. Above those VNFs, Data Collection Framework in Orchestrator is response of collecting status attribute. We have discovered that this work is not easy to accomplish due to the unfinished definition of monitoring API. Therefore, we propose a Data Collection Framework as below in Figure 1.
The monitoring data collection consists of three layers. The first layer is connecting between VIM and Orchestrator to collect resource pool status, like CPU usage, Memory usage, File-system usage and Nic(network) usage. The second layer is connecting between VNFM and Orchestrator to collect VNFs status. In addition, the third layer is connecting between NS manager and Orchestrator to collect service status.

2.2. Policy inventory

Conceptually, the policy is the key to trigger the network management action such as scale in/out. The analytic model will calculate the decision to hit the particular policies from policy inventory. For the policy in Orchestrator, previously there are simple pre-set policies, such as an attribute of a threshold value for VNF scaled up. When the status reaches the attribute threshold value, the system will trigger the policy action. However, the real situation of network is unstable or fluctuant. For example, we assume that the attribute value to trigger the scale-out action is by the CPU usage. The real situation is there are lots of peaks of CPU usage’s curve that over threshold value. How did the system decide to trigger the action of scaling out? With this question, we propose a closed circle to work out this problem, shown in Figure 2.
In Figure 2, the collection model depicts the data collection mentioned previously. The block of analysis depicts the analytic model described in the following section. It provides data analysis to predict the results or make decisions thus trigger the hit in policy inventory. The monitor block provides the feedback of the circle.

3. Analytic model and framework

As the development of Orchestrator, a more intelligent trend wishes to be researched and exploited. One of the FAPM’s function, analytic model, is designed for achieve the goal.

As tons of the status attribute data are collected, we cannot directly obtain the relationship among them. However, data analysis can be implemented to process and model the data to trigger the conclusion and decision-making to the orchestrator. Data mining is a particular data analysis technique that focuses on modeling and knowledge discovery for predictive [wiki]. It is a promising yet challenge direction to enforce into the Orchestrator.

The analytic module in the Orchestrator can be divided into two parts. One is the real-time analysis which can provide on demand network policy delivery for features such as dynamic scale in/out or auto-healing, etc. The other one is the non-real-time analysis that process batch data (e.g. log, etc.) and provide for static analysis and global resource planning. For real-time analytic model, we propose K-nearest neighbor (KNN) algorithm, because KNN can be realized in this analysis to assistant other components making real-time response. In non-real-time analytic model, unfortunately, the collection module has not accomplished, we just bring out an assumption of using decision trees learning, or neural networks.
3.1. The real-time analytic model

The real-time analytic model is a significantly dynamic part that is processing the real-time data and predicting results or trends for employing by other components, like optimized resource, work distribution, etc. It requires a real-time method to attain the target of quickly response and opportuniely updating model. Accordingly, the real-time KNN algorithm is proposed to realize the goal.

Firstly, KNN algorithm is a type of instance-based learning, or lazy learning used for classification. The input consists of the k closest training examples in the feature space, and the output depends on a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors. It assumes all the instances correspond to points in the n-dimensional space.

In the training phase, the algorithm only stores the training data and their corresponding labels. More precisely, an instance x can be described as

\[ [a_1(x), a_2(x), \ldots, a_n(x)] \]

where the \( a_i(x) \) denotes the ith attribute (e.g. CPU utilization or memory usage or file-system usage or nic usage or etc.) of instance x, and its corresponded label is marked as l. All kinds of labels will have their own corresponding relations to the following actions or events, like the number 1 could present to increase or decrease the number of virtual machines, etc.

In the classification phase, one defined number of k should be provided, and a testing instance y (or an unlabeled instance) is classified by assigning the label which is most frequent among the k training samples nearest to that query point [wiki]. Generally, the standard Euclidean distance is utilized to define the nearest neighbor of an instance as

\[ d(x,y) = \sqrt{\sum (a_i(x) - a_i(y))^2} \]

where \( d(x,y) \) denotes the distance between x and y. After obtaining the distance matrix D between the testing instance y and all training instances, the nearest k neighbors could be utilized to vote the label of the testing instance y.
Figure 3 illustrates an example of implementing the KNN algorithm, where the instances are points in a two-dimensional space and the labels are marked by different numbers. For example, the two-dimensional space could include two arbitrary attributes of collected data, like CPU usage and memory usage. The marked numbers could present the action of scaling up, scaling down or termination. The testing instance $y$ is shown as well. If the $k$ is assumed as 5, $y$ will be classified as class 1 by voting. Furthermore, this algorithm also can be implemented to analyze for either low-dimensional or high-dimensional data. It can be easily to extend the example of two-dimensional space to a higher dimension by increasing the number of attributes.

Figure 3: An example of KNN algorithm

Based on the KNN algorithm, our purpose is to achieve the real-time analytic model. Figure 4 describes the real-time model implemented in the KNN algorithm. Firstly, the same process is implemented to classify the testing data. In addition, there will be a detection model to check the label of it. If the label does not generate some anomalies, the testing data can be added into the training database to achieve the real-time analysis and update the training dataset.
KNN algorithm is one of the most fundamental and simple classification methods, which can be relatively easy to understand. In addition, it is a non-parametric method that can omit the process of setting and optimizing parameters. On the contrary, KNN algorithm is computationally expensive, and requires lots of memory. Fortunately, the orchestration usually implemented by cloud computing which can overcome the weakness. Moreover, there are other solutions to improve this algorithm, like dimensionality reduction, optimized training dataset, etc.

Eventually, the real-time analytic model is proposed. It is achieved by the real-time KNN algorithm to predict the outcomes or directions. Furthermore, the real-time analytic model can fulfill the dynamic control, which can assistant the Orchestrator to achieve a more intelligent way.

3.2. The non-real-time analytic model

For the non-real-time analytic model, it still plays an indispensable role to help the system to predict the future situation. It needs a pre-trained model to obtain the prediction of the following situation. In the case of that, it can help other components to achieve static analysis, etc. Therefore, the method of data mining (e.g. decision tree learning, neural networks, etc.) is a promising direction to deal with this.
4. Analytic framework in NFV Orchestrator

In this section, we would like to introduce the architecture of Orchestrator and how analytic and policy management (FAPM) modules fit in this Architecture. Since we initiate the Open-O project from mid-2015, we have finished most work in task fulfillment and monitoring. The architecture with analytic and policy is shown below in Figure 5.

Figure 5: The architecture with analytic and policy

From down to top, the Orchestrator supports Multi VIMs, such as OpenStack, as well as multi VNFM, such as tacker, Huawei VNFM and other 3rd party VNMFs. With help of those VNFM, we have onboard VNFs such as vEPC, vIMS, vBras, vCPE and Nanocell Gateway.

On the top of VIM and VNFM, the Orchestrator acts like a central brain of the network. There can be divided as three parts.
The first part is the FAPM module which include Data Collection Platform, Analytic and Policy Inventory. The Data Collection Platform is response for collection status form VIM, VNFM and NS which are discussed in section 2 and provides internal API for Analytic, Policy Inventory and Monitor module to use. The Analytic module is in charge of processing the network analysis and provide decision-making from the Policy Inventory. The Policy Inventory currently reposits policies which are input from NBI manually.

The second part of Open-O is Resource Orchestrator (OR) which includes Resource manager and Monitor. The resource manager is in charge of manager the connection with VNFM to globally manager the VNF and VIM resource. And the Monitor is called from the Data Collection Platform to get the system status.

The third part of Open-O is Network Service Orchestrator (NSO) which includes Model Designer, Catalog, NS lifecycle Manager and workflow engine. Model Designer can provide network operator and user a GUI that allows them to design the network topology and create data model of VNF, NS, VNFG, VL. The Catalog can store the VNF, NS, VNFG, VL data model. NS lifecycle manager is for NS level scale in/out.

Workflow engine assist the lifecycle management process by interacted with internal and external modules.

5. Security Considerations

6. IANA Considerations

7. Conclusions

In this draft, we have introduced data collection framework. And bring out an analytic model and framework with Orchestrator status data. At last, we introduced the analytic and policy architecture with Orchestrator.

8. References
8.1. Normative References


8.2. Informative References


9. Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot.

This draft is also contributed by Yuan Liu from China Mobile.

Authors’ Addresses

Vic Liu
China Mobile
32 Xuanwumen West AVE, Xicheng, Beijing
Email: liuzhiheng@chinamobile.com