vNMF: Distributed Fault Detection using Clustering Approach for Network Function Virtualization

Masanori Miyazawa and Michiaki Hayashi
KDDI R&D Laboratories, Inc.
2-1-15 Ohara, Fujimino, Saitama, JAPAN
{ma-miyazawa, mc-hayashi}@kddilabs.jp

Rolf Stadler
ACCESS Linnaeus Center
KTH Royal Institute of Technology
stadler @kth.se

***Abstract*—Network function virtualization introduces additional complexity for network management through the use of virtualization environments. The amount of managed data and the operational complexity increases, which makes service assurance and failure recovery harder to realize. In response to this challenge, the paper proposes a distributed management function, called virtualized network management function (vNMF), to detect failures related to virtualized services. vNMF detects the failures by monitoring physical-layer statistics that are processed with a self-organizing map algorithm. Experimental results show that memory leaks and network congestion failures can be successfully detected and that and the accuracy of failure detection can be significantly improved compared to common k-means clustering.**

Keywords— Fault detection; network function virtualization self-organizing map; in-network management

I. INTRODUCTION

With the emergence of new technology providing software-based telecommunication, which is called network function virtualization (NFV) discussed in the ETSI [1], the concept of incorporating its virtualization technology into the networking domain generally helps service providers reduce equipment, power consumption and accelerate the time-to-market for new services and network functionalities. To fully benefit from NFV, several challenges need to be overcome from a network management perspective. Fault detection in particular has become a top priority due to the increase in managed data and changed operation procedure for NFV.

Traditional networks are currently supported by a centralized management system such as an OSS (Operation Support System), which detect failures and analyze the root cause by integrally collecting management information, including statistical data and event data from network devices. However, under NFV, the amount of such managed data tends to keep increasing due to the expanded system scale and functionality with virtualization technology. The increased amount of managed data delays operational work, such as detecting failure and exploring affected services. The delayed failure detection, in turn, hinders network management activities, including the network recovery process, thus causing service quality to decline compared to the existing network. The challenge is to achieve scalable fault detection to support numerous virtualized infrastructures.

In addition, the existing network function usually comprises traditional network hardware appliances integrally provided by vendor-specific hardware, which are basically supported by a single vendor. In contrast, Virtual Network Functions (VNFs) are provided and supported by several vendors because the VNFs can share certain infrastructure resources such as hardware and hypervisor in NFV infrastructure (NFVI). In the complicated arrangements, the VNF is prone to unexpected and silent forms of failure, affected by failures from the separated NFVI. However, it is difficult to detect such failure under multi-vendor environments, since each VNF is supposed to have independent implementation of management interfaces. The fault detection must thus integrally monitor the tricky failures by monitoring performance behavior in unified NFVI.

It is challenging to promptly detect failures and performance anomalies in such NFV environments before they escalate into other failures. Many approaches to fault detection and prediction focused on large-scale network and cloud systems have been proposed. Most conventional approaches use supervised learning techniques [2, 3] to monitor production cloud systems. The supervised learning techniques can only detect previously known anomalies, but we assume increased detection of unknown anomalies in NFV because NFV components are basically unregulated structures comprising one-to-many relations between VNF applications and NFVI. As an unsupervised approach, clustering is one useful solution to detect anomalies in communication networks as well as wireless sensor networks. The author of [4] focuses on localizing failure for large-scale sensor networks and their localization technique in a distributed manner, which is implemented on each sensor node. The approach can effectively detect failures without known abnormal behavior, but requires a few metrics as input data and is not focused on complicated virtualization environments. Further work is still required in this area to validate applicability in support of NFV.

In this paper, we propose the design of a virtualized Network Management Function (vNMF) for NFV, which is achieved by our proposed two contributions; one of which is a the distributed detection framework to offload its function inside the network domain and analyze the huge number of statistical data in support of NFV for scalability. The other is a fault-detection mechanism implemented in vNMF, to detect abnormal failure on VNFs over NFVI statistical data at an early stage, using a Self-Organizing Map (SOM) algorithm. In addition, the proposed fault detection, which is implemented as a prototype vNMF system, was demonstrated for showing proof of concept. The results of the evaluation using memory-leak and network-congestion events event confirm that the proposed vNMF system successfully detected abnormal performance behavior.

II. MANAGEMENT FOR NFV

A. Problem statement for fault management on NFV

NFV allows network operators to deploy network functions such as firewalls, virtual routing and so forth on virtualized physical infrastructure, called NFVI, instead of traditional hardware appliances implemented on proprietary hardware. Its technology allows for various network services to be continuously created within a short time and many VNFs from different vendors for several dedicated network services to co-exist separately on the same NFVI, such as hypervisor and servers. The concept indicates that network structure and its operation would change dramatically compared to the existing network based on network hardware appliances. Amid such environment, VNFs would be prone to unexpected failure and impaired performance due to hidden failure (e.g., hardware failures, software bugs and traffic congestion failures) from separated NFVI. Accordingly, it is hard to understand the root cause of failure because the management framework for NFV, which is defined by NFV MANO [5], manages each component independently, while the Virtual Infrastructure Manager (VIM) and VNF manager are responsible for managing NFVIs and VNFs, respectively. For instance, if performance degrades due to CPU overload in an NFVI, the degradation may be detected by VIM. However, the event also affects other components such as virtual CPUs in NFVI , which means several alarms unsynchronized among NFVI, VM and VNF are generated respectively. The large number of alarms is sent to OSS to identify the root cause. However, the operation procedure for NFV is complex and takes considerable time; not only to detect failure but also identify the cause, given the lack of insight into the causal relationship of failures and its abnormal behavior in such NFV environments. With this in mind, services are at risk of degrading customer experience.

B. Requirements of fault detection function in NFV

To detect failure based on use cases, management functions must address three requirements for NFV:

[Req. 1] Detection of gray-box failure: One is to successfully detect unknown behavior and hidden failures in a virtualization environment, such as memory leaks, without knowledge of detailed VNF application-level information, because the failure information is often implemented as vendor-specific. It can provide advance notice of application-level failure before escalating into significant failures and eliminating the need for vendor-specific knowledge and prior knowledge of abnormal behavior.

[Req. 2] Scalability: The second is sufficient analysis resolution to detect faults because the operator wants to detect failures on a real-time basis and at an early stage before they escalate to critical failures. For instance, considering the management of a mobile network with the scale of a real telecom network, approximately tens of thousands of network functions including a control plane, data plane and packet switching function should be managed. In addition to this, we have to newly manage NFVI resources. In fact, the number of managed resources tends to increase compared to network hardware appliances. Scalable fault detection thus must be required to support the large-scale data monitoring and analyze the result.

[Req. 3] Short learning time: A short learning time is necessary because the life cycle of network and service in NFV tends to be shortened compared to existing telecommunication services. If we obtain training data set to cover all possible normal events and/or rare abnormal events for detection using a supervised learning algorithm over extended periods (e.g., over 1 month), the service and network configuration may already be changed, or the service may be terminated.

C. Failure model to detect faults on NFV

We assume that faults leading to failures on the VNF also affect performance metrics with NFVI (e.g., CPU usage memory usage, Disk I/O and network I/O). In paper [6], increasing failure instability emerges, whereby a number of interesting computer-related faults are verifiably preceded by a visible abnormal pattern. If we observe a behavior, taking various types of performance metrics into account and under normal and abnormal conditions, advance fault detection on NFV is possible using various types of performance metrics collected from NFVI. Figure 1 represents the potential impact on the relationship between resources when failures involving impaired performance occur in VNF and NFVI. Based on this assumption, no performance metrics with the VNF are required to detect the failure generated from the VNF by overall observation of performance behavior with NFVI.

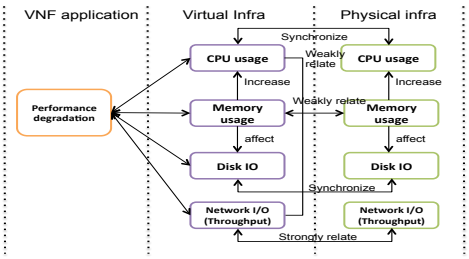

Fig. 1. Relationship of failure in NFV

III. VNMF FOR FAULT DETECTION

In this section, we present the design of a proposed vNMF as a scalable NFV management function, which can monitor statistical data of NFVI and detect failure by analyzing different type of performance metrics.

A. Fault detection mechanism

As described in Section II.C, to totally analyze the performance behavior of interrelated performance metrics, a clustering technique is useful to classify and extract relevant information from large data sources. One particular clustering method, known as the SOM [7], has several beneficial features that make it particularly useful for data clustering. SOM is a type of neural network and unsupervised learning algorithm. It is capable of analyzing various types of data and capturing its complex behavior while being computationally less expensive. It maps vectorial input data items with similar patterns on contiguous locations of a discrete low-dimensional grid of neurons in a manner that preserves topology. Its output data can be compared to a map: neighboring locations have similar data patterns attributed. Accordingly, SOM provides the analysis of various input data, lightweight and short learning time and is a prerequisite to our requirements.

The SOM comprises the following steps as shown in Fig. 2.

[Step 1]Initialization: All reference vectors m_i are initialized by random numbers. A sample is extracted from the training dataset and input vector x is input to all nodes simultaneously in parallel.

[Step 2]Competition: SOM identifies the node with the reference vector which most resembles the input data, called the best matching unit (BMU.) To calculate the BMU, each data is examined to find that with the reference vector most similar to the input vector. This selection is performed using the Euclidean Distance formula as equation (1), which measures similarity between the input data and the node with the reference vector.

$$\|x - m_i\| = \sqrt{\sum_{i=0}^{i=n} (x_j - m_{ij})^2} \tag{1}$$

The input vector x_j belongs to a winner node that is with the weight vector closest to x_j.

$$\|x - m_c\| = \min_i \{\|x - m_i\|\} \tag{2}$$

and defines the image of the input x on the map.

[Step 3]Adaptation: The nodes within a certain distance of the winning node c are updated according to the equation.

$$m_i(t+1) = m_i(t) + \alpha(t) h_{ci}[x(t) - m_i(t)] \tag{3}$$

h_{ci} is called the neighborhood function around the winner node c. Finally, SOM determines the stability condition. If the stability condition is satisfied, then the learning process terminates, otherwise go to Step 2 for another learning loop.

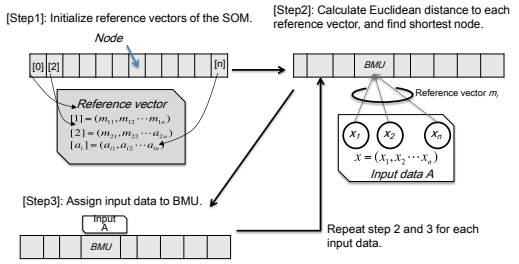

Fig. 2. The basic steps of SOM algorithm

After the input data is classified, some clusters must be divided to decide normal or abnormal behavior. In general, the number of clusters must be predefined in the case of k-means algorithm. The algorithms aim to minimize the sum of squared distance between all data points and the cluster center. The inconvenience of this procedure is the determination of the best value of k for optimal clustering. If the wrong number of k is set in advance, our results may be incorrect. However, it is rare to learn the appropriate number if unknown failure tends to increases. The author of [8] proposed a cluster-extraction algorithm using 1-dimensional SOM. To address the above problem, we apply a cluster-extraction algorithm based on a histogram to dynamically divide the number of clusters without pre-configuration. The main reason for choosing this clustering algorithm is its significant characteristic in SOM. When the input data comprises some clusters, the density of input data is high around the center of each cluster and low among the clusters. In the map built by SOM, the distance between two reference vectors assigned around the boundary of clusters exceeds that assigned around the center of each cluster. This characteristic allows clusters to be divided from input data by detecting valleys of the histogram of distance between the reference vectors of adjacency nodes. To do so, three processes can be used. First is the map building process, whereby the input data are subject to SOM for a set of reference vectors. The second is the map analysis process. For every node i, the reference vector density S (in other words, representing similarity between reference vectors) is found by distance between reference vectors from nodes i to i+1 using Euclidean distance.

$$S_i = \|m_i - m_{i-1}\| + \|m_i - m_{i+1}\| \tag{4}$$

Finally, based on node i, the cluster integration density is calculated using the following equation:

$$L_i = \frac{V_i}{S_i} \tag{5}$$

where V_i is the number of input data which is assigned to generate the histogram of L_i. Boundaries among the clusters can be extracted by detecting valleys of the histogram with an appropriate threshold L_{th} as shown in Fig. 3. Finally, to find a cluster concerning abnormal behavior in several separated clusters, a cluster covering minimum space is identified as the outlier cluster. Moreover, only if the identified cluster satisfies the following equation is it deemed to include abnormal behavior,

$$v_i \le \frac{v_h}{d} \tag{6}$$

where v_s is the coordinate number associated with the identified cluster, v_h is the total coordinate number concerning each cluster and d represents the number of the cluster.

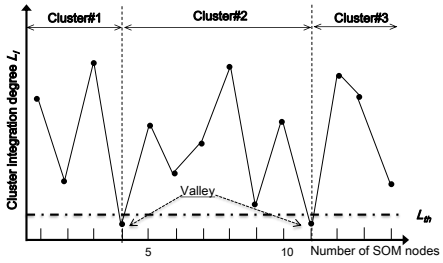

Fig. 3. Concept of cluster extraction

B. System design for vNMF

Figure 4 shows the system design of the distributed vNMF, which monitors statistical data, analyzes performance behavior and detects failure. The vNMF adopts a decentralized approach that is basically deployed on each NFVI and uses the residual resource to collect statistical data from VM and NFVI, process such data and detect faults. In our previous work, we proposed a distributed performance management framework [9], which offloads several performance management activities inside network devices to reduce the delay in comprehending network performance. To realize scalable fault detection, a fault-

detection function is additionally implemented in this framework as shown in Fig. 4. Firstly, the adapter collects the statistical data (e.g., CPU usage, memory usage, disk read/write I/O and in/out octets in/out packet, error packets as network I/O) from several VMs and NFVI via SNMP. After receiving data, the data is normalized using a data-processing function, due to the considerable variation among input data. Subsequently, the normalized data is stored into the DB function [10]. Finally, the fault detection function refers its data and analyzes by proposed clustering. If failure is detected, alarms are sent to OSS by the notification function.

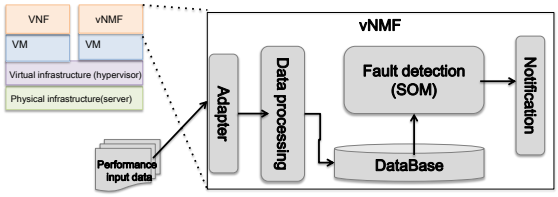

Fig. 4. vNFM system design

IV. PERFORMANCE EVALUATION

A. Evaluation methodology

To evaluate our proposed fault detection, a network testbed simulating memory-leak faults was configured as shown in Fig. 5. In this testbed, simulated NFV systems comprise physical infrastructure, such as the commodity server, virtual infrastructures, virtual machines and VNFs and have also implemented a vNMF prototype on each server. Table I summarizes the hardware and VM specifications for these experiments. The underlying interconnect is a 1 Gbps between physical NICs and 100 Mbps between virtual NICs. In terms of vNMF, it is responsible for collecting statistical data from physical and virtual infrastructure at 5-second intervals via SNMP. The data include CPU usage, memory usage, Disk I/O and network I/O respectively. Our proposed fault-detection function in vNMF was implemented using the R package [11].

TABLE I. HARDWARE AND VM SPECIFICATION

	CPU	Mem size	Software spec
Server 1	Intel (R) Xeon CPU @3.10GHz 4 core	8GBytes	- Cent OS 6.5 (host OS) - libvirt 0.10.2 - QEMU 0.12.1
VM-11 (VNF-app1)	2 core	2GBytes	- Cent OS 6.5 (guest OS) - Iperf 2.5.0.2 - Memory-leak program
VM-12 (VNF-app2)	2 core	2GBytes	- Cent OS 6.5 (guest OS)
VM-13 (vNMF)	2 core	2GBytes	- Cent OS 6.5 (guest OS)
Server 2	Intel (R) Xeon CPU @3.10GHz 4 core	2GBytes	- Cent OS 6.5 (host OS) - vNMF(R package)
VM-21 (VNF-app3)	2 core	2GBytes	- Cent OS 6.5 (guest OS) - Iperf 2.5.0.2
VM-22 (VNF-app4)	2 core	2GBytes	- Cent OS 6.5 (guest OS) - Neo4J 2.0.4
VM-23 (vNMF)	2 core	2GBytes	- Cent OS 6.5 (guest OS) - vNMF(R package)

To demonstrate the effectiveness of the proposed fault detection, we evaluate it based on two remarkable use cases, such as both memory-leak and network congestion. We assume that it is hard to understand the events under VNF applications running under vendor-specific knowledge.

[Use Case #1]: We evaluated a real-world memory-leaking event using an open source graph database [12] for comparison with the use case of VNF (Use case #2). The DB system was implemented in VNF-app12. The memory-leak event was generated by creating data and deleting it periodically within a single transaction process [13]. The memory leak ran out of available memory size in approximately two hours.

[Use Case #2]: The next use case is a memory-leak event on a simulated VNF, performed to simulate network traffic between VNFs using traffic generators installed into both VNF-app11 and VNF-app21. In addition, a memory-leak event was simulated by a simulated memory-leak program on the VNF-app11. As with use case#1, the memory leak ran out of available memory in two hours.

[Use Case #3]: We also evaluated network congestion internally generated in server 2. Bi-directional 80Mbps traffic was simulated between VNF-app21 and -22 using traffic generators. To generate traffic overload, additional traffic was fed from VNF-app22 to VNF-app21. The traffic slowly increased for 20 minutes.

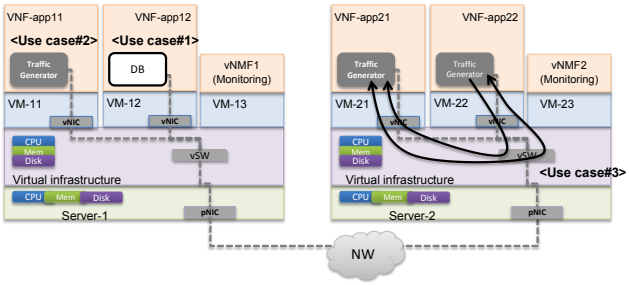

Fig. 5. Experimental setup

B. Evaluated results of the fault detection

We initially evaluated the effectiveness of the SOM-based fault detection approach. Figure 6 shows the training data put into SOM. The training data are all normal condition data and represent no rapid change in each performance behavior. Figure 7 shows each input data retrieved from VM-12 and server 1 when a memory-leak failure by DB bug occurs in VM-12 (Use case #1). After the memory-leak event occurs, the available memory size gradually decreases and it took approximately two hours to run out of available memory. After two hours, it eventually started to use swap space. Furthermore, disk I/O as well as CPU usage fluctuated considerably due to the increased swap usage. Conversely, CPU usage and disk I/O with server 1 also rose after 120 minutes in response to the significant fluctuation in VM-12. In the case of simulated VNF (Use case #2), the observed performance behavior basically resembled variations with use case # 1.

Figure 8 (a) depicts the total number of faults detected per ten-minute period by vNMF1 based on use case #1. Although it detected fewer than 30 failures on average before 60 minutes, there is a significant jump in the rate of reported failures around 60 minutes. We consider this increase due to the variation in available memory size on VM-12 and the variation in CPU usage on server 1 compared to normal conditions. Toward 120 minutes, which is the time when available memory runs out, the rates of increase converged. The graph represents

the trend of proportional increase in association with the progression of memory leak. Note that vNMF can detect a maximum of 12 fault detections per minute because it analyzes fault detection at 5-second intervals. In this evaluation, we identified the occurrence of abnormal behavior symptoms concerning memory-leak when vNMF detected more than 60 failures on average within ten minutes. Accordingly, we confirmed the memory-leak failure manifested after 60 minutes. To demonstrate adaptability other than for memory-leak events, proposed fault detection under network-congestion events (use case #3) was evaluated. Figure 8 (b) depicts the total number of faults detected per 5 minutes by vNMF2. The result shows a significant jump in the rate of reported failures after 5 minutes in association with the progressive network congestion on server 2. Thus, we observed that traffic losses occurred on the virtual switch after around 10 minutes, which confirmed that our approach was working properly, since more than 30 failures were detected in around 10 minutes. Accordingly, we confirm the ability of our fault-detection method to adapt to both memory leaks and network-congestion events.

In addition, to understand the efficiency of the proposal and reveal the difference of the detection rate between use case #1 and #2, we conducted sensitivity experiments to study how vNMF performed under different parameters associated with SOM clustering. Three important parameters decisively influenced the fault-detection rate. One was the number of SOM nodes with the potential to affect fault-detection accuracy, as shown in Fig. 9 (Left graph: use case #1, right graph: use case #2). For fewer than 13 nodes, there was no false positive but the number of true positives was fewer than 13 nodes. Conversely, for more than 13 nodes, the number of false positives soared. Accordingly, the result shows that 13 nodes are reasonable to detect abnormal behavior. Comparing the case of an actual DB application (Use case #1) with that of a simulated VNF (Use case #2), the results were subject to the same tendency, but the number of true positives of use case #2 was smaller than that of use case #1. In the case of use case #2, no outstanding variation in CPU usage was observed on server 1 compared to use case #1. Accordingly, the difference between use case #1 and #2 is considered to be greatly influenced by CPU usage behavior with server 1. Likewise, we evaluated the impact of the number of training data as depicted in Fig. 10. Given the large volume of training data, there were few true positives and when it was small, considerable false positives. From the result, 11000 training data, equivalent to a training window of approximately one hour, was required to detect faults more accurately. In comparison with both use case #1 and #2, the results were subject to the same decreasing tendency regarding the number of true- and false positives. Finally, we verified the impact of the threshold value L_{th} to decide the number of clusters as shown in Fig. 11. The nature of the threshold value means if a large value is set, the number of false positives rises, given the increased number of divided clusters. The result indicated that the number of false positives had increased in both use cases as the threshold value exceeded 0.05 and we confirmed that around 0.03 was an optimal value. Based on the results of Figs. 8 (a), 9, 10 and 11, we confirm that this evaluation indicates the effectiveness of our fault-detection method for memory leaks generated from not only actual DB applications but also simulated VNF, and outperforms the baseline approach where direct measurements are analyzed.

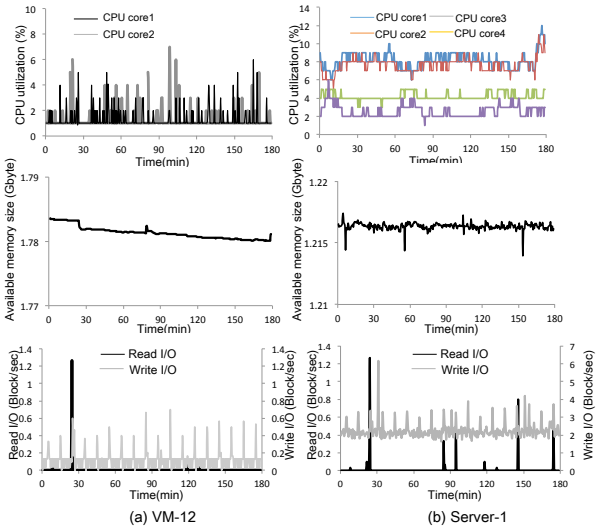

Fig. 6. Training data under normal condition

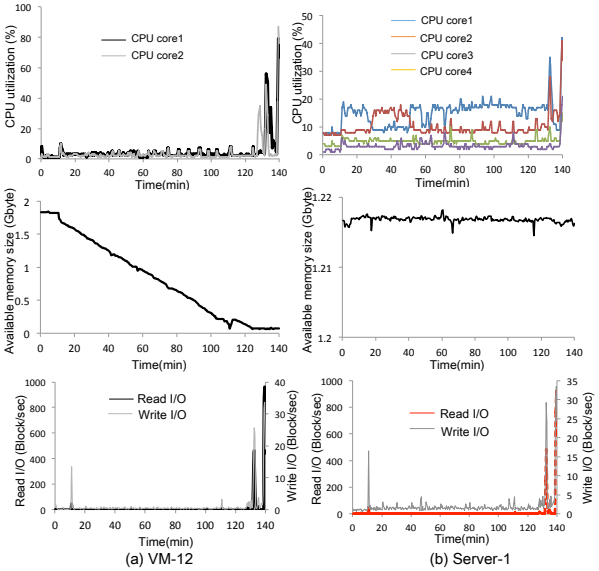

Fig. 7. Performance behavior in VM-12 and server 1 during memory-leak (Use case #1)

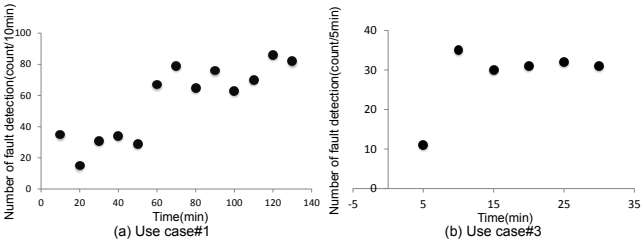

Fig. 8. Total number of detection in the case of use case #1 and #3.

Finally, to evaluate the accuracy of our fault detection, we compared our approach using SOM to k-means clustering.

Figure 13 shows a comparison result under memory-leak event (Use case #1). The comparison experiment was performed using the same training data depicted in Fig. 6 and the same input data shown in Fig. 7. We used a Receiver Operating Characteristic (ROC) curve to compare the performance, which can effectively show a tradeoff between the true positive and false positive rates, respectively. The ROC curve for SOM was obtained by adjusting the value of L_{th}. For the k-means meanwhile, the ROC curve was calculated by adjusting the distance threshold value, which decides the boundary cluster to judge normal or abnormal data. As a setting of k-means, the k was set to 20. The choice of k value used in this experiment was based on five trial experiments conducted with k set to 5, 10, 15 and 20, respectively. The performance of k-means-based detection represented no significant improvement when k was set to a value exceeding 20. As shown in Fig. 12, the significant best result showed a 97% true positive rate along with a 0.2% false positive rate. We confirm that the result indicates that the performance of our approach could achieve higher accuracy than common k-means clustering.

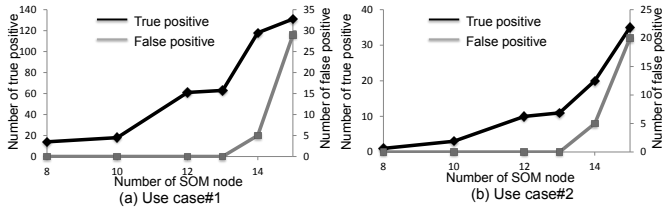

Fig. 9. Relation between the number of SOM nodes and the detection rate

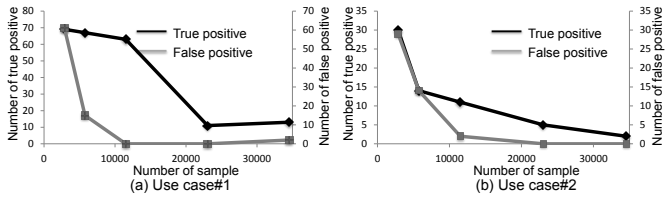

Fig. 10. Relation betwenn the number of training data and the detection rate

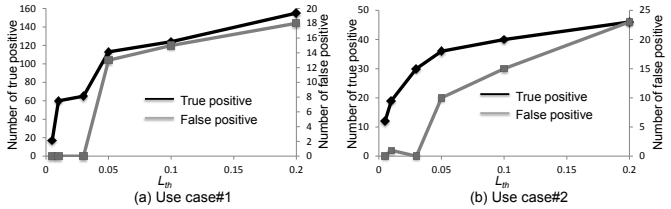

Fig. 11. Relation between L_{th} and the detection rate

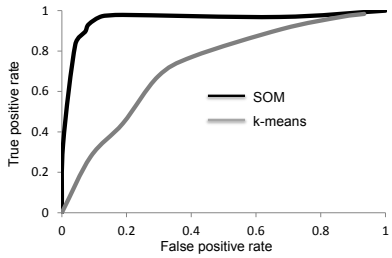

Fig. 12. Accuracy comparison between SOM and k-means clustering

V. Conclusion

In this paper, we proposed a distributed network management function for NFV called vNMF, which was achieved by our two contributions; one was a distributed fault detection framework to offload its function inside the network domain to analyze the huge number of statistical data in support of NFV for scalability, the other was a fault detection mechanism. The detection mechanism identified abnormal behavior generated from VNF application at an early stage by integrally analyzing various types of statistical data with NFVI using SOM-based clustering. Finally, we evaluated the effectiveness of our proposed fault detection using a prototype of the vNMF system. The results showed memory-leak failure and network congestion failure, which is complicated in terms of finding the failure on virtualization, were successfully detected using only one-hour training dataset, and its accuracy detection was significantly improved compared to the common k-means clustering approach. These results also showed that it met Req.1 (Detection of gray-box failure) and Req.3 (Short learning time) that are our requirements for fault detection described in Section II.B Regarding Req.2 (scalability), we indicated the design of a distributed framework that can deploy the fault detection on each server. In our future work, we plan to need to evaluate different scenarios to demonstrate effectiveness of the proposed fault detection, and complement the scalability evaluation with the vNMF by comparison with common centralized approach. Accordingly, the proposed vNMF is expected to facilitate scalable network management toward more complex network virtualization environments.

References

[1] Network Function Virtualization Reference Architecture (ETSI GS NFV 002 V1.1.1)

[2] A.W.Williams, S.M.Pertet, and P.Narasimah, “Tiresias:Black-box Failure Prediction in Distributed Systems,” In Proc. of IPDPS 2007, pages 1-8, Mar. 2007.

[3] Y. Tan, H. Nguyen, Z. Shen, X. Gu, C. Venkatramani and D. Rajan, ”PREPARE: Predictive Performance Anomaly Prevention for Virtualized Cloud Systems,” in Proc. of ICDCS2012, Pages 285-294, Jun. 2012.

[4] L. Paladina, M. Paone, G. Jellamo, and A. Puliafito, “Self organizing maps for distributed localization in wireless sensor networks,” in Proc of ISCC 2007, pages 1113-1118, Jul. 2007.

[5] ETSI ISG NFV GS NFV-MAN 001 V0.6.1 Network Function Virtualization (NFV) Management and Orchestration.

[6] G. T. A. Dumitras and P. Narasimhan, “Fault tolerance and the magical 1%,” In ACM/IFIP Conference on Middleware, pages 431-441, Grenoble, France, Nov. 2005.

[7] T. Kohonen, “Self-Organizing Maps,” Springer-Verlag Berlin Heidelverg, 1995.

[8] M. Terashima, et al., “Unsupervised Cluster Segmentation Method Using Data Density Histogram on Self-Organizing Feature Map,” (in Japanese), Transactions of the Institute of Electronics, Information and Communication Engineers, Vol.J79-D-II, No.7, pages.1280-128, 1996.

[9] M. Miyazawa, M. Hayashi, “In-network real-time performance monitoring with distributed event processing,” In Proc. of MANFI 2014, pages 1-5, May. 2013.

[10] Apache. Cassandra-2.0.8: http://cassandra.apache.org

[11] R project: http://www.r-project.org

[12] Neo4j 2.0.4 : http://neo4j.com

[13] http://neo4j.com/release-notes/neo4j-2-1-3/