Verification of Openstack Operation with Normality-degree of Workflows

Ryota Mizutani, Takeshi Usui, and Yoshinori Kitatsuji
KDDI Research, Inc. 2-1-15 Ohara, Fujimino, Saitama, 356-8502 Japan
E-mail: {ry-mizutani, ta-usui and kitaji}@kddi-research.jp

***Abstract*— Verifying job executions in operating virtual networks has emerged as a crucial issue for network operators. Now, tracking and monitoring log messages of the operations is a typical approach. However, a monitoring system to detect a critical operation is based on text matching with the previously obtained log messages indicating normal operations. It basically has a limitation that similar but different log message flows become indeterminable to judge the normality of the operation. We propose a solution that estimates the normality-degree for the indeterminable log message flows. We evaluate the difference in normality-degrees between normal and abnormal completion message flows. We show that the proposed method distinguishes these cases more clearly than the conventional method. This indicates that the proposed method enables the system administrator to classify job execution into three types (i.e., normal, semi-normal, and abnormal completions). As a result, the proposed method allows the system administrator to focus on critical job executions in troubleshooting operations.**

Keywords-NFV; Openstack; Log message; Normality-degree; Similarity-degree

I. INTRODUCTION

The rapid spread of virtualization technology [1] has motivated network operators to procure communication service systems consisting of network function virtualization (NFV) [2][3]. NFV allows network operators to deploy network functions on the virtualized infrastructure where the physical entities (switches, routers and servers) provide a virtual (logical) network composed of logical links, routers and servers (called virtual links, routers and machines, respectively). In general, the logical entities are realized by software programs. In addition, the network topology of the virtual network is not identical to that of the physical entities that compose the individual logical entities.

Openstack [4] maintains the virtual networks (i.e., to create, initiate, terminate and delete virtual machines and to manage the virtual network with them). Because the major network providers (AT&T [5], Deutsche Telekom AG [6], Rackspace Cloud [7], and so forth) have adopted Openstack, it is the de-facto standard for virtual network management systems in commercial.

Openstack is composed of multiple function blocks (e.g., Neutron, Nova, and Cinder described in Section II), and the function blocks are composed of multiple service processes, e.g., an interface to other function blocks, a scheduler for tasks initiated by arrival requests, and individual major processes. The service processes run independently (in a parallel manner) in the function block.

Openstack executes jobs, e.g., creates, initiates, terminates and deletes virtual machines, with these entities. To verify whether the jobs are normally executed or not, the system administrator tracks the log messages produced by the service processes. However, this procedure is often complicated, because Openstack inherently assigns no job identifier to the log messages, and the series of log messages obtained by the system administrator become scrambled over the function blocks. Then, it is difficult to verify job completions by simply tracking the log messages.

Regarding the result of job executions, there are three types of normality judgements: normal, semi-normal, and abnormal completions. A normal completion implies that the job completes successfully without any problem. A semi-normal completion implies that the job eventually completes although there are some problems, typically, a long delay. An abnormal completion implies that the job actually does not complete as a designed procedure. Regarding Openstack, each kind of job completions has various sequences of log messages (referred to as workflows). It is because the orders of function blocks and service processes vary, and consequently the orders of their log messages result in varying.

A typical judgement of job executions involves matching the current workflow with previously given workflows (referred to reference workflows) with normal, semi-normal and abnormal completions. The system administrators cannot comprehensively assess all the workflows. Therefore, the monitoring of job executions (workflows) eventually encounters indeterminable judgements in the case where the workflow does not match any of those previously given. We refer to this workflow as an indeterminable workflow.

To identify an indeterminable workflow in the normality judgements, several similarity algorithms have been used. The text matching rate is used to identify an indeterminable workflow to the reference workflow [8]. Text matching utilizing the TD-IDF is employed to diagnose anomalous patterns [9], and cosine similarity is used to detect traffic anomaly deviations [10]. These researches identify the indeterminable workflow in the most similar unique reference workflow. Though, these methods disables to normal identification if there are no reference workflow (i.e., unknown anomaly).

To tackle this identification issue, we propose a method that utilizes similarity estimation adopted in string-pattern matching. The similarity estimation rates differ between two strings expressed by three differences types: addition, deletion, and displacement of string portions. We express a log message as a symbol of a string, and a workflow as a string. We estimate the similarity between workflows in

 ISBN: 978-1-61208-574-6

order to express the normality-degree of the job execution. We evaluate the difference of normality-degrees between log message flows with normal, semi-normal and abnormal completions. The result reveals that the proposed method can distinguish the above three cases more clearly than the conventional method.

The rest of the paper is organized as follows. In Section II, we introduce the issues associated with normality estimation regarding the log messages in Openstack. In Sections III and IV, we propose and evaluate the method of normality estimation. In Section V, we conclude this paper.

II. ISSUES ENCOUNTERED WHEN ESTIMATING FAULTS FROM A WORKFLOW

This section gives an overview of Openstack and workflows, and the issues encountered in the operational flow with Openstack, especially in the area of troubleshooting with accumulated indeterminable job executions.

A. *Overview of Openstack and Workflows*

Openstack has multiple function blocks that manage virtual resources (e.g., memory and CPU) provided by the physical entities. The primary function blocks are Nova, Neutron, Cinder, Swift, Glance, Keystone, and Horizon. Nova builds the virtual machines and routers [8]. Neutron sets up links between the virtual machines and routers and manages basic configurations (e.g., IP address assignments, and topologies). Cinder manages block storages for virtual machines and routers. Swift manages the configuration templates for setting up the multiple virtual machines simultaneously. Glance provides OS images for virtual machines and routers. Keystone provides an authentication service corresponding to the job executions. Horizon provide user interfaces, typically with graphical user interfaces.

Each of the functions has single or multiple dedicated service processes. The service processes run dedicated tasks and each task produces log messages. For instance, when the virtual system administrator commands the creation of a virtual network, a command signal originates from Horizon (the user interface), and is sent to Keystone (for authentication), Nova and Cinder (building virtual machines), Glance (installing OS), and so forth.

Generally, Openstack is composed of several physical or virtual servers. These servers have different roles. For example, the controller server has a management role and requests job execution to the control servers. The computer server provides the resources (CPU, memory, storage, and so forth) with a virtual machine.

A major reason for workflow variations is that some function blocks execute multiple service processes, and request tasks to the other function blocks in parallel. This leads to addition, deletion, and displacement of portions of log messages in the workflow. Fig. 1 shows samples of a variety of workflows. In this figure, the characters represent a single line log message. These three patterns are the most

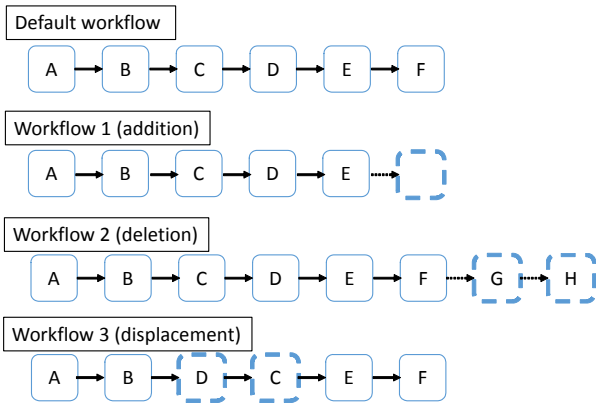

Figure 1. Samples of a variety of workflows

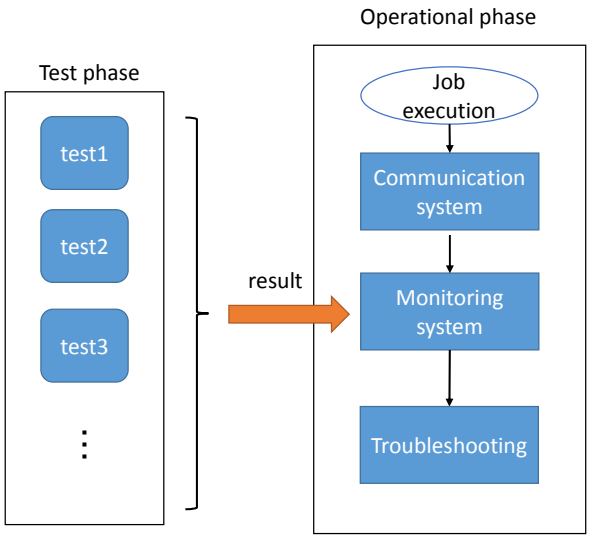

Figure 2. Flow of communication system operation.

typical pattern variations. We assume the default workflow is the most frequent pattern that is generated when executing a job. One is addition. In workflow1, the character “F” needs to be added to correspond the default workflow. Another is deletion. In workflow2, the character “G” and “H” need to be deleted to correspond to the default workflow. The other is displacement. In workflow3, the character “C” and “D” need to be displaced to correspond to the default workflow.

Fig. 2 illustrates the generic flow of communication system operation. The communication system has the feature of continuously providing its communication services. The system operational phase follows the test phase.

The test phase improves the completion of development before the communication system starts to provide its communication services. This phase comprehensively tests all the designed processes of the system, (or as many as possible) to remove bugs remaining from the time of its development. Some of the tests should be for all job executions, and are used for the monitoring system in the operational phase.

 ISBN: 978-1-61208-574-6

TABLE I. JUDGEMENT CASES PROVIDED BY MONITORING SYSTEM ESTIMATING NORMALITY WITH WORKFLOWS PREVIOUSLY GIVEN: SIMILAR/DIFFERENT PATTERN TO/FROM THEM.

		Workflows previously given	
		Workflows with normal completion	Workflows with abnormal completion
Judgement of a new job execution	Similar pattern	Case *A*	Case *B*
	Different pattern	Case *C*	Case *D*

In the operational phase, however, they are initially not enough because of the workflow variations. We conjecture that a monitoring scheme takes exact-matching of workflows. This results in indeterminable completions occurring when monitoring job executions in the operational phase. This would tend to be a large number, especially in the initial period. As the operational phase proceeds, the indeterminable workflows should decrease as a result of providing the judgement results (normal, semi-normal and abnormal completion) revealed by troubleshooting of the monitoring system. The given workflows and their judgements are referred to as reference workflows, hereinafter.

A significant issue is that it takes a long time to remove potentially critical elements in the communication system. Actually, it is not clear which indeterminable completion the system administrator should focus on among the accumulating completions, in troubleshooting operations. They may include critical job executions, and a system failure will occur before long. Our challenge is to prioritize them according to the criticality of the indeterminable workflows.

Table I illustrates the judgement cases and the reasons that provided by monitoring system. This is based on the job executions with workflows previously given. The columns are types of execution results belonging to workflows previously given: normal and abnormal. The rows represent the similarity/difference to/from the given workflows under the assumption that the monitoring system can estimate normality.

When the monitoring system judges that there is a new job execution with indeterminable completion, there are four possible combinations in terms of given workflows with normal or abnormal completion: AB, AD, CB, and CD. Case AB implies that the workflow of a new job execution is similar to the workflows previously given with both normal and abnormal completions. In this case, one of the workflows previously given could be wrong, or they may just be similar (in other words, this implies that normality estimation is difficult). In the case of AD or CB, the results are probably correct, i.e., the results are likely to be normal or abnormal completions in the cases of AD or CB, respectively. Case CD implies that the monitoring system has identified no effective workflow preliminarily given.

It is desirable that troubleshooting for indeterminable completions first deals with case CB (almost certainly abnormal), and cases AB (one of the given workflows is possibly wrong), CD (no reference to estimating normality), and AD (almost certainly normal) in this order. Our challenge is how to estimate similarity degree among multiple workflows previously given, in order to prioritize the job executions with indeterminable completions for troubleshooting. The next section describes the proposed method that solves this challenge.

III. PROPOSED METHOD

In this section, we propose a solution for estimating normality-degree for the indeterminable workflow of job execution. We first give an overview of definition of normality-degree, second, an algorithm for normality-degree is presented, and third, a method for judging normal, semi-normal and abnormal completion is described.

A. Overview of Normality-degree Estimation

We provide an overview of the process employed in the proposed method. First, the system administrator collects the reference workflows (how the jobs have been completed: normal, semi-normal and abnormal completions) from the job execution results. Second, the proposed method uses a string-similarity evaluation method (evaluation of addition, deletion and displacement of two strings) in order to measure the similarity of workflows between the indeterminable workflows and reference workflows. Third, the normal distribution of the similarity of workflows is calculated in order to distinguish normal, semi-normal and abnormal completions more clearly. Finally, the proposed method estimates the normality-degree using the normal distribution, as described in Section III-C.

The normality-degree represents how closely the indeterminable workflows correspond to the reference workflows. As described in Section II-C, the system administrator uses the normality-degrees as a reference to select a particular job execution with indeterminable completion from among many for troubleshooting.

B. Proposed Method Expressing Similarity of Reference Workflows

This subsection describes the preliminary process in three steps: workflow construction, similarity measurement, and forming a similarity distribution regarding normal, semi-normal and abnormal completions.

1) Workflow Construction

In the test and operational phase, the workflows are obtained from the continuous generation of log messages. In the test phase, the log messages are compiled while a series of tests are conducted. In the operational phase, the system administrator performs multiple job executions. First, the system administrator must decide how to divide log messages into multiple sets of messages according to the individual job executions

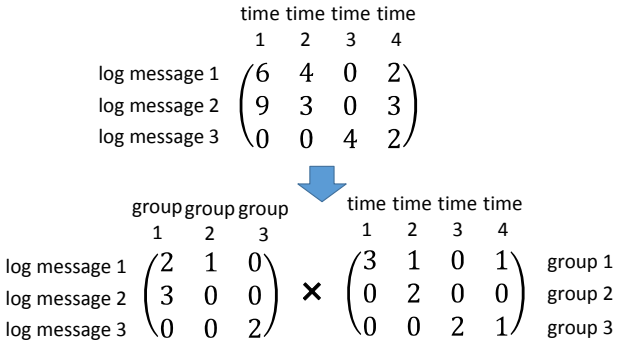

Figure 3. Sample of NMF utilization

TABLE II. RELATIONSHIP BETWEEN LOG MESSAGE TYPE AND CORRESPONDING CHARACTERS

Log message type	Corresponded character
INFO nova.osapi_compute.wsgi.server [admin admin] ... "GET /.//flavors/ HTTP/." status: len: time: .	A
INFO nova.osapi_compute.wsgi.server [admin admin] ... "GET /.//images/ HTTP/." status: len: time: .	B
INFO nova.compute.claims [admin admin] [instance:] Claim successful	C
INFO neutron.agent.securitygroups_rpc [None None] Refresh firewall rules	D
INFO neutron.plugins..drivers.openvswitch.agent.ovs_neutron_agent [None None] Configuration for devices up [] and devices down [] completed.	E
INFO nova.compute.manager [None None] [instance:] VM Started (Lifecycle Event)	F
INFO nova.compute.manager [None None] [instance:] VM Paused (Lifecycle Event)	G
INFO nova.compute.manager [None None] [instance:] During sync_power_state the instance has a pending task (spawning). Skip.	H

The naive method to divide log messages generally uses the identifier assigned to the log messages indicating which job execution produces [9][10]. However, Openstack assigns a unique identifier to log messages related not to a job execution but a function block. This implies that the log messages can only reveal which function block causes a problem when log messages indicate an error or warning. We utilize a non-negative matrix factorization (NMF) [11][12] that divides log messages into multiple groups. This method can divide a mixture of data sets with counter attributes into multiple groups by reasonably distributing the values of counters into the formed groups [13]. The proposed method utilizes NMF to obtain two benefits: one is dividing log messages into workflows, and the other is deriving the number of workflows. Fig. 3 gives an example of results obtained by utilizing NMF. The original matrix presents the number of log messages for every time unit (e.g., every ten seconds). NMF divides this matrix into two matrixes: one is the number of log messages belonging to the workflow (denoted by 'group' in Fig. 3), and the other is the number of workflows (groups) in every time unit. In dividing a matrix, the original matrix is represented by the two smaller matrices, which, when multiplied, approximately reconstruct original matrix. To apply NMF to the log message, we use Table II to classify the log messages depending on the contained messages and service block IDs, job types, and signal types.

NMF still needs a heuristic approach to provide a substantial number of data sets divided into the particular groups. In the proposed case, we give the number of workflows sequentially, and find the appropriate number of the workflows where the difference between the original matrix and the multiplication of two divided matrixes (the summation of the mean square error of matrixes' elements) is below a threshold obtainable by scanning the number of workflows.

2) Similarity Measurement

After obtaining the reference workflows, the proposed method tries to find any trends in workflow similarity. Specifically, it estimates the degree to which sequences of log messages are identical in terms of their length, and displacement.

The proposed method takes the concept from the similarity-degree evaluated by Jaro distance [14], which measures the string similarity based on the number and order of the common characters. It is generally utilized in order to identify typos and spelling mistakes in sentences. This algorithm computes a similarity score normalized between 0 to 1 where 0 indicates no similarity and 1 indicates an exact-match. Jaro distance D between two strings X and Y (difference between ASCII codes) is defined as:

$$D = \begin{cases} 0 & \text{if } m = 0 \\ W_1 \times \frac{m}{L_X} + W_2 \times \frac{m}{L_Y} + W_3 \times \frac{m-t}{m} & \text{otherwise} \end{cases} \quad (1)$$

where m and t denote the number of matched characters, and the number of displacements, respectively. L_x and L_y denote the character lengths of strings X and Y, respectively. Three coefficients, W_1, W_2, and W_3, are weights in the following ranges: $0 \le W_1 \le 1$, $0 \le W_2 \le 1$, $0 \le W_3 \le 1$, and $W_1+W_2+W_3=1$.

The first and second terms compute ratios of common characters in terms of strings X and Y. The third term computes the ratio of displacement against the common characters. No matter where the common characters are positioned in the strings, it is counted as m. For example, when two strings X='abcd' and Y='abdec' are given, the similarity score D is about 0.77 in the cases of W_1, W_2, and W_3 for 1/3. The first and second terms of (1) can be seen as the ratios of common characters in terms of strings X and Y.

When strings X and Y are identical, the first and second terms, respectively should be 1. When string X is longer than string Y (this implies that string X has more uncommon characters more than string Y), the first and second term should be low and high, respectively. Additionally, when the strings X and Y are switched, the values of first and second terms also switch

In order to apply this to workflows, first, we utilize the correspondence table shown in Table II in order to denote the workflows as the strings. Second, we treat three terms of (1) separately, as follows:

$$D_1 = \begin{cases} 0 & \text{if } m = 0 \\ \frac{m}{L_X} & \text{otherwise} \end{cases} \quad (2)$$

$$D_2 = \begin{cases} 0 & \text{if } m = 0 \\ \dfrac{m}{L_Y} & \text{otherwise} \end{cases} \quad (3)$$

$$D_3 = \begin{cases} 0 & \text{if } m = 0 \\ \dfrac{m-t}{t} & \text{otherwise} \end{cases} \quad (4)$$

where variables X, Y, m, t, L_X, and L_Y are the same as those used in (1). Equations (2) and (3) evaluate the ratios of common log messages in terms of the two workflows X and Y, and (3) evaluates the ratio of displacement of the common characters. In the following discussion, we refer to D_1, D_2, and D_3, as addition, deletion and displacement scores, respectively.

3) Expressing the population of workflows

The step in this subsection derives the probability density regarding the normal, semi-normal, and abnormal completions in the reference workflows to express the population of the reference workflows. First of all, we surveyed the workflows obtained from various job executions, and found that almost all the reference workflows are different from each other, but there are many similar patterns of log messages for each of normal, semi-normal or abnormal completions. Additionally, there are few pairs of workflows that are markedly different from the similar log message patterns of workflows. Therefore, we conjecture that a normal distribution can reasonably express the population of the workflows.

Actual computations are used to derive the normal distributions for the addition, deletion, and displacement scores for each of the normal, semi-normal and abnormal completions. For this, the proposed method computes the averages and variances of the addition, deletion and displacement scores. These probability densities are used to estimate the normality-degree of the indeterminable workflow that occurs in the operational phase, as described in the next subsection.

The computation of the averages and variances takes all the pairs of workflows in the reference workflows. Because the addition and deletion scores can switch when the compared workflows switch, these scores should have an identical average and variance. Each normal distribution for addition, deletion and displacement is represented by:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \quad (5)$$

where $f(x)$ is a function of probability density, μ is the average of the similarity scores, and σ^2 is the deviation of the similarity scores. Fig. 4 shows some examples of three probability densities for the addition (D_1), deletion (D_2) and displacement (D_3) scores for workflows with normal completion.

This derivation of probability density is conducted for each of the semi-normal and abnormal completions in order to estimate the normality-degree, as described in the following subsection

C. Estimation of Nomality Degree

In order to estimate normality-degree, we propose products of the derived probability density from the addition, deletion and displacement scores.

The naive approach can be employed to utilize the similarity-degree measurement algorithm directly. However, it has some difficulties when two or more differences occur in the similarity computation where two or all the terms in (1) are affected. Actually, no pair of workflows are identical, and all the terms become less than 1, and similarity D of (1) cannot clearly show the difference.

We show the steps for estimating the normality degree. First, we collect the indeterminable workflows and calculate the three types of similarity between the reference workflow and indeterminable workflow derived from (1)-(3) in the operation phase. This step is the same as that done in the test phase described in section III-B. Next, we obtain the probability density by substituting the similarity into a normal distribution (5). Finally, the proposed method (products approach) simply multiplies each probability density derived from (5) for the addition, deletion and displacement scores. We regard this multiplied value as the normality degree. As shown in Section IV-B, the results clearly showed the difference (or fitness) to the judgement of the indeterminable workflows. We take probability density into account because we found that almost all the reference workflows are different from each other, but there are many similar log message patterns for each of the normal, semi-normal and abnormal completions. Additionally, there are few pairs of workflows that are markedly different. Therefore, we conjecture that the normal distribution can reasonably express the population of the similarity.

IV. EVALUATION

This section evaluates the distinguishability of the normal, semi-normal and abnormal completion workflows identified using the proposed method by comparing the conventional approach with the product approach. First, we describe the environment from which we obtained the log messages; secondly, we show the evaluation result.

A. Evaluation Environment

We obtained the log messages from the evaluation environment. We adopted Mitaka [15] as the version of Openstack, and set up a single pair of controller, computer and log servers. Each server has Ubuntu 16.04 as its operating system, a CPU with 12 cores, and 32GB of memory. The log server collects all the log messages generated in the controller and computer servers, and merges them into a single file.

In order to construct the reference workflows, we selected “VM creation” as an example of job execution. The created VM was assigned a single virtual CPU and 2 GB of memory. A hundred VM creations were included in the reference workflows.

To evaluate the performance of our proposed method, we prepared ten indeterminable workflows for the three use cases, as follows.

 ISBN: 978-1-61208-574-6

TABLE III. MAXIMUM AND MINIMUM NORMALITY-DEGREE WHEN USING COSINE DISTANCE

	use case 1	use case 2	use case 3
max	1.000	0.897	0.632
min	0.917	0.688	0.381

TABLE IV. MAXIMUM AND MINIMUM NORMALITY-DEGREE WHEN USING JARO DISTANCE

	use case 1	use case 2	use case 3
max	1.000	0.944	0.769
min	0.827	0.750	0.481

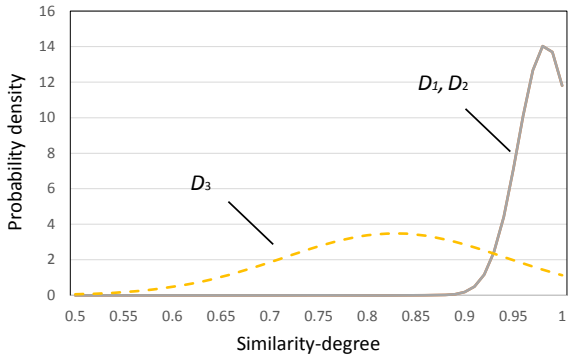

Figure 4. Distribution of probability density of normal workflow

[Use case 1] Normal completion: This is an example of normal completion of job executions.

[Use case 2] Overload: This is an example of semi-normal completion of job executions. When one function block in Openstack sent a signal to another block, we made a message queue with excessive traffic, and it causes signal losses.

[Use case 3] Network down: This is an example of abnormal completion of job executions. We shut down the interface of the computer server connected to the controller server. That interface was mainly used to send Openstack operational signals.

B. *Evaluation result*

We verified the performance of the proposed method described in Section III. First, we show the similarity-degree between the reference workflow and the three use cases of indeterminable workflows. Table III shows the maximum and minimum normality-degree when using cosine distance. Table IV shows the normality-degree when using Jaro distance (1) with coefficient for 1/3 is used. We define this similarity-degree calculation method as the conventional approach. Our purpose is to distinguish among the three use cases. If the difference in the normality degree of the two use cases is large, there is a high possibility that the two use cases can be distinguished. Tables III and IV show that the difference for each use case is low, and it confirms that it is difficult to distinguish the use cases. Next, we derived the probability density (normal distribution) as shown in Fig. 4. The addition and deletion scores (D_1 and D_2) have an identical shape, as described in Section III-B. Their shapes are sharp, which implies that almost all the workflows have the same number of common log messages. On the other hand, the displacement score (D_3) has a blunt shape, which implies that the patterns of displacement vary.

TABLE V. MAXIMUM AND MINIMUM NORMALITY-DEGREE WHEN USING PRODUCT APPROACHES

	use case 1	use case 2	use case 3
max	1.000	0.567	7.02E-05
min	0.919	0.110	1.33E-09

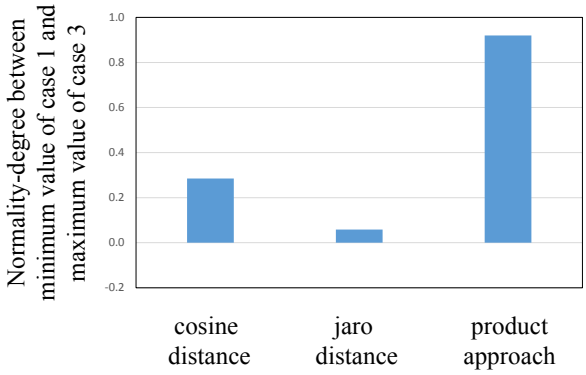

Figure 5. Normality-degree between minimum value of case 1 and maximum value of case 3

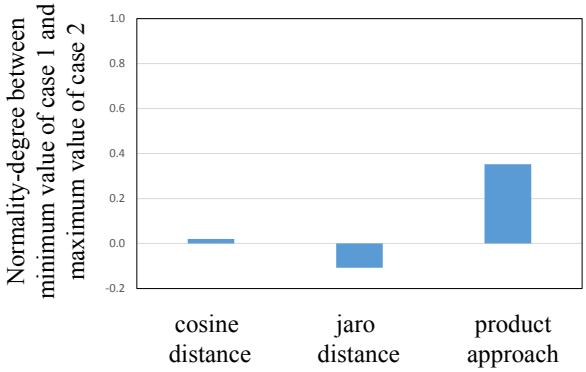

Figure 6. Normality-degree between minimum value of case 1 and maximum value of case 2

Table V shows the maximum and minimum normality-degree, computed by the product approach. The results were calculated from the probability density (normal distribution) and normalized between 0 and 1. In the case of the product approach (proposed method), (2), (3) and (4) were used and derived three normal distributions as shown in Fig. 4. Compared with Tables III and IV, we found that the value of use cases 2 and 3 becomes lower due to the application of normal distribution.

Fig. 5 shows the normality-degree between the minimum value of use case 1 and the maximum value of use case 3. Fig. 6 shows the normality-degree between the minimum

 ISBN: 978-1-61208-574-6

value of use case 1 and the maximum value of use case 2. This normality-degree indicates the distinguishability of the two use cases. Generally, use case 1 shows higher similarity than the other use cases. Therefore, the difference between the minimum value of use case 1 and the maximum value of other use cases is an important criterion for distinguishing among the use cases.

The results for the conventional and product approaches are clearly different. The product approach gives a higher normality-degree than the conventional approach does. This effectively works to distinguish between the normal and overload samples of the indeterminable workflows. In the conventional case, it is difficult to distinguish between the normal and overload samples. However, in the product approach, the normality-degrees have over a 0.3 difference between the normal and overload samples.

Although this evaluation only included the normal completions of the reference workflows, the semi-normal and abnormal completions show similar results to this normal completion case. We conjecture that this is because the proposed method takes into account the product of the addition, deletion and displacement. The product emphasizes low similarities, and makes clear the differences in the normality-degrees. This allows the system administrator to easily distinguish the indeterminable workflows that need troubleshooting.

V. CONCLUSION

This paper proposed a verification method for job execution in Openstack. The proposed method estimates the normality-degrees which are referred to in order to detect indeterminable workflows among many workflows (workflow with the least abnormal degree should be first).

The proposed method has two benefits: one is a way to divide log messages accumulated through multiple job executions into individual workflows with NMF; and the other is a way to estimate the normality-degree. The latter uses string similarity evaluation as a reference. We disassemble the string addition, deletion and displacement scores, and define the normality-degree by multiplying them.

We also revealed that the workflows in Openstack had similar but, strictly speaking, various patterns, and the proposed normality-degree has the ability to clearly distinguish the workflows.

The following issues remain as future work: evaluation of semi-normal and abnormal completions of the reference workflows, and clarifying the degree to which our approach (troubleshooting according to the normality-degree) brings stability to virtualized communication systems.

REFERENCES

[1] B. Pfaff, J. Pettit, T. Koponen, K. Amidon, M. Casado, and S. Shenker, “Extending networking into the virtualization layer,” in Proc. of work shop on Hot Topics in Networks (HotNets-VIII), 2009.

[2] ETSI GS NFV 002: Network Functions Virtualisation (NFV); Architectural Framework. [Online]. Available from: http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.02.01_60/gs_NFV002v010201p.pdf 2017.06.10

[3] B. Vazirnezhad, F. Almasganj, and S. M. Ahadi, “Hybrid statistical pronunciation models designed to be trained by a medium-size corpus,” Computer Speech and Language 23, pp. 1-24, 2009

[4] Openstack. [Online]. Available from: https://www.openstack.org/ 2017.06.10

[5] AT&T. [Online]. Available from: https://www.openstack.org/videos/video/at-and-ts-cloud-journey-with-openstack 2017.06.10

[6] Deutsche Telekom AG. [Online]. Available from: https://www.telekom.com/en/media/media-information/enterprise-solutions/telekom-strengthens-the-openstack-community-with-the-open-telekom-cloud-363316 2017.06.10

[7] Rackspace Cloud. [Online]. Available from: https://www.rackspace.com/cloud/private 2017.06.10

[8] B. C. Tak,S. Tao, L. Yang, C. Zhu, and Y. Ruan, “LOGAN: Problem Diagnosis in the Cloud Using Log-based Reference Models,” In Cloud Engineering (IC2E), pp. 62-67, 2016

[9] P. Zhou,B. Gill,W. Belluomini, and A. Wildani, “GAUL: Gestalt analysis of unstructured logs for diagnosing recurring problems in large enterprise storage systems,” In Reliable Distributed Systems, pp. 148-159, 2010

[10] C. B. Jiang,I. Liu, Y. N. Chung, and J. S. Li, “Novel intrusion prediction mechanism based on honeypot log similarity,” International Journal of Network Management, 2016

[11] Open Virtual Switch. [Online]. Available from: http://openvswitch.org/ 2017/06/10

[12] Q. Fu, J.-G. Lou, Y. Wang, and J. Li, “Execution anomaly detection in distributed systems through unstructured log analysis,” In Proceedings of the 2009 Ninth IEEE International Conference on Data Mining, ICDM ’09, pp. 149–158, 2009.

[13] K. Kc and X. Gu. Elt, “Efficient log-based troubleshooting systemfor computing infrastructures. In Reliable Distributed Systems(SRDS),” IEEE Symposium, pp. 11–20, 2011.

[14] D. D. Lee and H. S. Seung, “Learning the parts of objects with nonnegative matrix factorization”, Nature, 401, pp.788–791, 1999.

[15] D. D. Lee and H. S. Seung, “Algorithms for non-negative matrixfactorization,” Advances in Neural Information Processing Systems 13, MIT Press, pp. 556–562, 2001.

[16] T. Kimura, K. Takeshita, T. Toyono, M. Yokota, K. Nishimatsu, and T. Mori, “Network Failure Detection and Diagnosis by Analyzing Syslog and SNS Data: Applying Big Data Analysis to Network Operations,” NTT Technical Review, Nov 2013, Vol. 11, No. 11

[17] W. E. Winkler, "String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage," Proceedings of the Section on Survey Research Methods. American Statistical Association, pp.354–359, 1990.

[18] Opentack Mitaka. [Online]. Available from: https://www.openstack.org/software/mitaka/ 2017/06/10

 ISBN: 978-1-61208-574-6