A Workflow-less Autonomous System Architecture for Assurance Operations using Choreography Architecture

Kensuke Takahashi
NTT Network Service Systems Laboratories
NTT Corporation
Tokyo, Japan
kensuke.takahashi.gm@hco.ntt.co.jp

Tomoki Ikegaya
NTT Network Service Systems Laboratories
NTT Corporation
Tokyo, Japan
tomoki.ikegaya.ya@hco.ntt.co.jp

Ryota Katayanagi
NTT Network Service Systems Laboratories
NTT Corporation
Tokyo, Japan
ryota.katayanagi.wt@hco.ntt.co.jp

Satoshi Kondoh
NTT Network Service Systems Laboratories
NTT Corporation
Tokyo, Japan
satoshi.kondou.hm@hco.ntt.co.jp

Tsuyoshi Toyoshima
NTT Network Service Systems Laboratories
NTT Corporation
Tokyo, Japan
tsuyoshi.toyoshima.dk@hco.ntt.co.jp

***Abstract*— In this paper, a workflow-less system architecture adopting Choreography architecture is proposed that can flexibly and quickly respond to changes in demand on a system. This approach makes each maintenance function into a microservice and enables each microservice to operate autonomously in an event-driven manner. In addition, a label assignment method is proposed in the communication control between the maintenance functions. With the proposed method, a series of operations can be automated without defining a workflow. Experiments were carried out using the proposed method, and results show that a workflow with synchronous control could be achieved without a workflow definition.**

Keywords— Autonomous Distributed System, Network Management, Operation, Choreography Architecture

I. INTRODUCTION

According to the Information Technology Infrastructure Library (ITIL) [1], which details the best practice in IT service management, the purpose of service operation is to offer a service at the level agreed with the customer. Site Reliability Engineering (SRE) [2] proposed by Google summarizes concepts and best practices to achieve both fast development speed and Service Level Objectives (SLOs), treats availability tolerance as an error budget, and aims to keep systems and services available at all times within the error budget. In the case of services that use networks or clouds, service monitoring, fault detection, analysis, and restoration are conducted by using judgment and system tools on the basis of the knowledge and know-how of the maintainers in order to meet the agreed service level.

978-3-903176-31-7

However, the diversification of services and business models that flexibly combine network functions such as Service Function Chaining (SFC) [3], Network Functions Virtualization (NFV) [4], and Cloud-native Network Functions (CNF) [5] (e.g., the business to business to X (B2B2X) business model as known as wholesale) has made operations more complex and diverse, resulting in an increase in maintenance operations. For this reason, workflow engines such as StackStorm [6] and Rundeck [7] are used to automate operations to reduce maintenance operations such as routine fault detection, analysis, and recovery, as well as maintenance operations such as deployment and configuration. These automation efforts involve defining the workflow by the connections between the parts for processing and judgment. However, with the improvement of agility in response to the recent changes in the business environment and the shortening of the time to market, the changes in requirements for the system (e.g., adding new operational capabilities or managing operations) are becoming intense. As a result, workflow definitions created for automation tend to become larger and more complex, and it is difficult to evaluate how changes in the environment, such as changes in the specifications of services and equipment to be managed, will affect the parts and connection relationships defined as workflows, and how changes in specific parts and connection relationships will affect other parts and connection relationships. We propose an operation support system architecture based on Choreography architecture. In this system, the operation functions constituting the operation flow such as information collection, information analysis, test, and recovery for the

management objective are made into microservices (workers), and the workers are unconnected and autonomously operate in an event-driven manner in accordance with the information output from other workers. Thus, by eliminating the workflow definition, the operation is automated, and the changes in the request to the system quickly responded to.

In this paper, we describe the existing research on the automation of operations in Chapter 2, the purpose of this paper in Chapter 3, and the tasks for achieving a workflow-less system in Chapter 4. After the proposed method is described in Chapter 5, the experimental results are described in Chapter 6, summarized in Chapter 7, in which future problems are also described.

II. Related works

A. Workflow Automation

To automate a task, the workflow and the decision rule (method for acquiring and comparing information to be used) need to be clear, and it is necessary to arrange IF-THEN rules and procedures for using various system tools. The operations of monitoring life and death of specific processes on network (NW) equipment and servers and monitoring abnormalities by using resource information such as CPU/memory and thresholds are automatically carried out by network management systems such as Zabbix [8]. In addition, a technology has been proposed to unitarily and automatically carry out processing to construct a service by combining infrastructure services individually provided by multiple enterprises such as Infrastructure as a Service (IaaS) and Virtual Private Network (VPN) [9]. In addition, a Network Management System (NMS) [11] has been proposed that is based on Active Information Resources (AIR) [10], which is a concept enabling active cooperation by adding meta information, etc. to information. The AIR-NMS classifies the maintainers' knowledge of trouble management into three categories: knowledge about the set of symptoms and candidate causes, knowledge about the identification of the cause of the trouble, and knowledge about the cause of the trouble and the countermeasure plan. By presenting the cause and the countermeasure plan to the maintainers, support can be provided for trouble management.

B. Microservice architecture

Microservice [12] architecture has been proposed for the development and operation of IT systems. In this architecture, each microservice 1) runs as a separate process and 2) communicates primarily over the network and processes certain tasks. Each microservice can be started, deployed, and updated independently of other microservices. In this microservice architecture, there are two patterns that connect the local transactions of individual microservices to form a flow: Orchestration and Choreography. Orchestration has an Orchestration program that controls the entire processing flow, executes the service in response to a request from the Orchestration program, returns the execution result as a response, and takes over the next processing "request reply system." Choreography is a method in which each microservice is operated autonomously in accordance with the setting of operating conditions and which of the services given to each microservice in advance starts next, and a specific microservice is often driven by the occurrence of some event and becomes "event-driven system."

The Orchestration architecture has disadvantages such as the difficulty in detecting and responding to processing that occurs at a time that the Orchestration program does not know or an event that occurs at an unexpected timing, and the processing cannot be performed or the timing is delayed. However, there is a claim that the Orchestration architecture can cope with complex workflows. On the other hand, the Choreography architecture has a merit that the processing timing is not delayed because it can immediately notify the subsequent microservices when an event occurs. However, it has difficulty responding to a workflow that requires processing in a complicated order because each microservice operates autonomously.

III. Objective of this work

We aim to establish a system architecture that can respond flexibly and quickly to changes in system requirements, while reducing the number of maintenance personnel by automating operations. Here, the change in the request to the system refers to the addition of an operation function or an object to be managed. As mentioned in the previous section, many operations are automated using orchestrators and workflow engines. However, the existing automation is achieved by mechanically defining the work procedure in the operation flow, which has been performed by maintenance personnel until now, as a workflow, in which a specific alarm is triggered, an event is separated and dealt with, and finally the restoration confirmation is performed. The workflow consists of two parts for work and judgment and connections between the parts.

Though the work and judgment to be carried out by each component and the connections between components are simple, the workflow definition tends to be large and complicated because the operation flow includes many operations and conditional branches. Therefore, there is a problem on following up the change in the demand. That is, it is difficult to discern the parts and connections that are affected when the update of the equipment as a management object and the service specifications change, and it is also difficult to discern how the changes of parts and connections affect the whole workflow.

Therefore, we propose an architecture that adopts the Choreography architecture of microservices, turns each maintenance function such as information collection and analysis into a microservice (hereinafter referred to as a worker), and enables a series of operations to be conducted in a workflow-less manner by each operation autonomously in an event-driven manner.

IV. CHALLENGES

To be able to respond flexibly and quickly to changing demands on the system, two requirements must be satisfied.

Requirement 1. To automatically operate a series of operations without a workflow and asynchronously by appropriately issuing an event by each worker.

Requirement 2. To add new workers or change existing ones without affecting other workers

To satisfy Requirement 1, if the choreography architecture is adopted and workers autonomously work with each other in an event-driven manner, the orchestration program that controls the workflow is no longer required and can be implemented. For communication between workers, there are many existing technologies such as Advanced Message Queuing Protocol (AMQP) [13] and Apache Kafka [14] as asynchronous messaging protocols and middleware. However, as described in Requirement 2, even if a new worker is added or an existing worker is modified, in order not to affect other workers, it is necessary to reduce the dependency between the workers as much as possible and to make them loosely coupled.

A. Loose coupling requirements among workers

To make workers loosely coupled, the following requirements must be satisfied.

Requirement 2-1. A worker can issue a message without holding information of other workers.

Requirement 2-2. A worker can decide whether or not to execute independently regardless of other workers.

Requirement 2-1 is required when each worker issues its own processing result as an event. If this requirement does not exist (e.g., information about other workers such as an IP address is required to issue a message), each time a new worker is added, an effect occurs, and it becomes impossible to quickly respond to a change in a request to the system. Therefore, a worker needs to be able to issue a message without holding information of other workers.

Requirement 2-2 means that each worker can determine whether or not the worker needs to execute a service (e.g., AWS-EC2) without referring to the internal processes of other workers. When there is a fault analysis worker (worker B) whose input is the information acquired by a dedicated monitoring worker (worker A), if worker B cannot decide whether or not to execute worker A without referring to internal information such as whether or not worker A has completed the information acquisition, the modification of worker A affects worker B. In this way, if workers A and B are tightly coupled, a function must be added, etc. to a certain worker from the investigation of the affected worker, as in the case of the workflow definition, and it becomes impossible to promptly respond to the change in the demand for the system. Therefore, the worker needs to be able to determine whether or not execution is necessary independently of other workers.

V. PROPOSED METHOD

We propose a workflow-less method using the choreography architecture to satisfy Requirements 1 and 2. To satisfy Requirement 1, a messaging broker function is required for asynchronous communication of messages between workers. There are many existing technologies for this function. To satisfy Requirements 2-1 and 2-2, it is necessary to consider a communication method between workers using a messaging broker utilized to satisfy Requirement 1. We propose a method for managing the rules of message destinations by assigning abstracted information to each worker as a Label.

A. Label assignment method

Since the IF-THEN rule uses information specific to the worker, such as the message content issued by each worker, the addition of a function to the worker affects other workers. Therefore, we propose a label assignment method for solving the problem by determining the necessity of execution of each worker using abstracted information rather than information specific to the worker.

In this method, attribute information is given to each worker as a label, and when a message is transmitted, the label information given to the worker itself is given to the message. Then, the transmission destination is controlled by the label information given to the message.

As a result, the worker serving as the producer can issue a message without holding the information of the worker serving as the consumer, so Requirement 2-1 can be satisfied. In addition, since the transmission destination is controlled on the basis of the label information given to the message, it becomes multicast instead of broadcast, and the message can be transmitted only to the consumer who needs the message information.

In this method, each worker can limit the message to be received only to the message to which the label required by the worker is attached. Therefore, even if a function is added to the worker, the attribute information “label” is not changed, so Requirement 2 -2 can be satisfied.

To achieve this method, three features are required: 1. the ability to manage label assignments and message destination rules for workers, 2. the ability to label messages, 3. the ability to control messages on the basis of destination rules. Details of each function will be described below.

1) Ability to manage labels assigned to workers and rules for message destinations

A label is given to all workers as attribute information. Then, a label assignment rule is provided for managing which label is assigned to each worker. A message rule for managing a rule for determining a transmission destination of a message from a label of a transmission source imparted to the message is provided. The label assigned to the worker can be determined by the label assignment rule. Further, in accordance with the message transmission rule, it is possible to determine a

destination label to which a message should be sent and a worker to which the destination label is assigned.

2) How to Label Messages

When a worker transmits a message, the worker refers to a label assignment rule, assigns label information to the message to be transmitted with the label information assigned to the worker, and transmits the message. It is possible to transmit a message utilizing the label.

3) Method for controlling messages based on label and destination rule

A message broker is provided with a message control function for controlling the destination of a message. The message control function obtains the label information of the transmission destination by referring to the message rule, sets all workers to which the label is given as a destination from the label information, and transmits the message.

As a result, a message can be sent to an appropriate worker by setting the destination of the message.

VI. EXPERIMENT

To evaluate the feasibility of the proposed method, an experiment was carried out using RabbitMQ [15], which is a type of message control software.

Fig. 1 shows a configuration in the experiment. In this experiment, the message destination rules are managed by using the Topic method of Exchange in RabbitMQ, and routingKey is treated as Label. In the Topic method, the Exchange sends the message to the Queue that matches the routingKey to the Queue of each worker. When a message is transmitted, the label assignment function in the message transmission and reception section of the worker adds the routing key information to the transmitted message and transmits it to the message, and the message control of the proposed method becomes possible. In this experiment, the process type of maintenance operation, which is the minimum classification, is adopted as a concrete label type in order to confirm that the effect to other workers caused by the addition and repair of workers is minimum by the control using the label. That is to say, there are six kinds of labels: UI, monitor, test, shaping, analysis, and recovery. When a worker transmits a message, the worker acquires the label information assigned to the worker from the label management information and assigns the label information to the message as the routingKey information. For example, when the test Worker E transmits a message, information of "routingKey" = "test" is added to the message. The Exchange of RabbitMQ determines whether the sent message matches the routingKey set in Queue and determines the destination. "By this control, the message to which "routingKey" = "test" is given is distributed to Queue to which Worker G of information processing is connected. Thus, even when a new worker is to be added, only the label information of the worker itself is examined. For example, when a worker for information collection is newly added to achieve an operation for a new service, message transmission can be controlled by the Topic system of Exchange if a label, i.e., "routingKey" = "monitor" is provided as label management information.

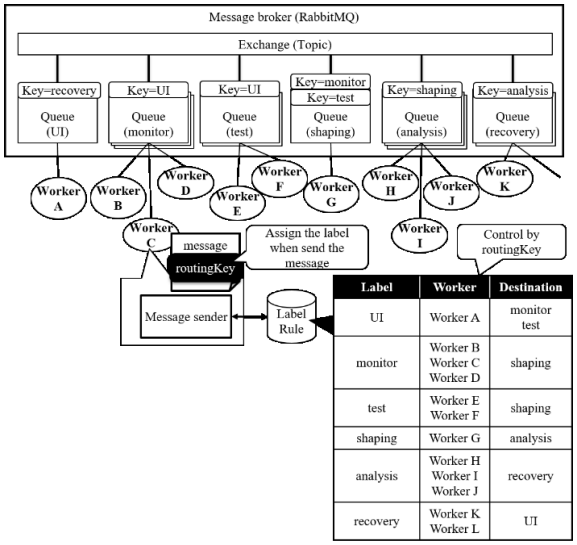

Figure 1 Experiment configuration

On the other hand, the worker who receives the message tries to execute its own processing because the message is sent to the Queue that is connected. At that time, the maintenance operation is done by autonomously judging whether the processing such as whether the data necessary for the action processing of each worker is ready or whether the parse processing is possible.By this experiment, it was possible to realize the message control as expected, and the feasibility of this proposed method was confirmed.

VII. CONCLUSION

In this paper, we proposed a workflow-less system architecture based on the Choreography architecture that can respond flexibly and quickly to changes in system requirements. To achieve this, each maintenance function was made into a microservice, and each function operated autonomously in an event-driven manner. To ensure loose coupling among maintenance functions, we proposed three methods: a method to manage a label assigned to a worker and a rule of a message destination, a method to assign a label to a message, and a method to control a message on the basis of the label and the rule of a message destination. We demonstrated the feasibility of the proposed method by the operation verification using RabbitMQ, and even when a new worker is added, it is possible to incorporate the proposed method only by examining the label information of the worker itself. In the future, by machine learning the autonomous operation and workflow-less communication of each worker, the normality of the flow of a series of operations is monitored, and when there is an error in the operation of each worker, a technique capable of self-repairing is examined. Through these efforts, we aim to realize AI-OpS that can adapt itself to changes in the environment.

REFERENCES

[1] ITIL, Foundations of IT Service Management with ITIL 2011

[2] B. Beyer, C. Jones, J. Petoff, and N. R. Murphy. Site Reliability Engineering: How Google Runs Production Systems. "O'Reilly Media, Inc.", 2016.

[3] Service Function Chaining (SFC) Architecture, http://www.rfc-editor.org/info/rfc7665

[4] ETSI GS NFV 001 V1.1.1, "Network Functions Virtualisation(NFV); Use Cases", https://www.etsi.org/deliver/etsi_gs/NFV/001_099/001/01.01.01_60/gs_NFV001v010101p.pdf

[5] Cisco, "Cloud-Native Network Functions (CNFs) White paper", https://www.cisco.com/c/en/us/products/collateral/routers/cloud-native-broadband-router/white-paper-c11-740841.html

[6] StackStorm, https://stackstorm.com/

[7] Runeck, https://www.rundeck.com/open-source

[8] Zabbix, https://www.zabbix.com/jp/

[9] N. Oguchi, X. Wang, P. Palacharla and T. Ikeuchi, "Seamless Network Service Orchestration across On-Premises and Cloud Infrastructures," 2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Berlin, 2017, pp. 1-2, doi: 10.1109/NFV-SDN.2017.8169863.

[10] Y. Takahashi et al., "Knowledge Oriented Network Fault Resolution Method Based on Active Information Resource," 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, Toronto, ON, 2010, pp. 361-364, doi: 10.1109/WI-IAT.2010.123.

[11] T. Hoshino, Y. Tanimura, K. Sasai, G. Kitagata and T. Kinoshita, "Support mechanism for the collaboration between humans and agents in network management tasks," 2017 IEEE 6th Global Conference on Consumer Electronics (GCCE), Nagoya, 2017, pp. 1-2, doi: 10.1109/GCCE.2017.8229428.

[12] J. Lewis and M. Fowler, Microservices - a definition of this new architectural term, https://martinfowler.com/articles/microservices.htmlt.

[13] AMQP, https://www.amqp.org/

[14] Apache kafka, https://kafka.apache.org/

[15] RabbitMQ, https://www.rabbitmq.com/