A Self-Adaptive Measurement Rate Control Method for an Agent-based Service Monitoring System

Masaru Sakai
NTT Network Service Systems Laboratories
NTT Corporation
Tokyo, Japan
masaru.sakai.pf@hco.ntt.co.jp

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

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

***Abstract*— The increasing complexity of services of telecommunications carriers has increased the burden of operators monitoring these services and the load on the system monitoring the target devices. Operators need to choose appropriate monitoring settings for each component of the services, considering the properties of each component. In addition, whenever a component is added or changed, its monitoring settings need to be reconfigured. Furthermore, if operators configure incorrect monitoring settings, service anomalies may not be detected, or quality of service may be damaged. Therefore, the monitoring system needs to automatically input and change the monitoring settings in accordance with the service. In this paper, we propose a method to dynamically adjust the monitoring interval for a monitoring system. Finally, evaluation results show the proposed method can reduce the monitoring overhead more effectively than the previous method and can also ensure the accuracy of monitoring when a service abnormality occurs.**

Keywords— Software engineering, self-adaptive, Monitoring rate, Agent-based monitoring

I. INTRODUCTION

Most services of telecommunications carriers are required to continue to operate without stopping. To sustain these non-stop services, their resources must be monitored. Accurate and timely updates of resource statuses are necessary to quickly detect failures and promptly restore service functions.

Agent-based monitoring systems are often used for monitoring service resources. In agent-based monitoring, agents (*i.e.*, software for data gathering) are installed in devices to be monitored (*e.g.*, physical servers, virtual servers, docker containers). When a manager (*i.e.*, software for data aggregation) requests data from each agent, the agents transmit the latest resource statuses to the manager.

Examples of widely used open source software for agent-based monitoring are Zabbix [1] and Prometheus [2]. Generally, in agent-based monitoring, a manager extracts information from agents on the basis of a fixed monitoring rate. Therefore, if the monitoring rate is too low, abnormalities in services can be missed, and if the query rate is too frequent, the network bandwidth owned by the monitoring software can be enlarged or storage capacity for accumulating information will be insufficient. However, in the distributed computing environment, the manager must monitor an enormous number of objects, thus the overhead of monitoring is significant. Therefore, for example, in [3], the optimal fixed monitoring interval that keeps the balance between the CPU load and the consistency is determined by experiments.

Methods have been proposed to adaptively adjust monitoring rates for network traffic [4, 5, 6, 7, 8], . These methods aim to adapt the interval of requesting to network devices on the basis of network traffic behavior. Especially, Self-tuning Adaptive Monitoring (SAM) [4] achieves an appropriate tradeoff between the accuracy and the overhead of monitoring by adopting Additive Increase Multiplicative Decrease (AIMD) for adjusting a window function used to calculate a new monitoring interval. However, previous approaches tend to respond to constant small fluctuations in the monitored data and frequently narrow the monitoring period. Many common types of service resource data (*e.g.*, CPU usage, memory usage, network Tx, network Rx) contain constant noise, thus, in terms of the service assurance, previous methods may not effectively reduce the overhead of monitoring. Therefore, to achieve the right balance between accuracy and overhead in service resource monitoring, an adaptive monitoring interval adjusting method is required that is robust to stationary noise.

In this paper, we propose a novel adaptive monitoring rate control method for service assurance that is less susceptible to constant noise in service resource data. Our method uses cosine similarity to measure the degree of change in time series data, and the monitoring rate is increased when an significant sudden change occurs that is different from previous trends. We create sample data simulating the time series of service resources and execute the proposed method and SAM [4] on the sample data. The sample data includes normal parts and anomaly parts where the value increases and decreases assuming a failure in the service. Through a comparative evaluation, we show that our approach can keep the monitoring interval wide in the normal part to reduce the overhead of monitoring and shorten the interval in the anomaly part to ensure accuracy more significantly than SAM [4].

The remainder of this paper is organized as follows. Section II provides detailed background and explanations of previous approaches. Section III details the proposed method step by step. Section IV explains how to generate time series data to execute the proposed method. Section V describes the experiment configuration and evaluation results. Section VI concludes the paper.

II. BACKGROUND

This section describes the importance of service monitoring, recent architectures of services, the marketed products used for service monitoring and the previous research into adaptive monitoring rate control.

A. Service Assurance

Services of telecommunications carriers are often required to stay on 24 hours a day, 7 days a week. Leaving a service failure obligates a carrier to pay a penalty determined by a service level agreement (SLA), which can be huge loss to revenue. Thus, carriers generally form specialized teams to perform 24-hour service assurance.

Service assurance can be divided into six processes: monitoring, analysis, problem isolation, recovery policy decision, recovery execution, and confirmation of recovery. These processes are sequentially processed starting from monitoring when there is a service failure. Therefore, the accuracy of the monitoring process is important because it affects all five subsequent processes. That is, the monitoring process must be able to collect fine-grained information when a service anomaly occurs.

B. Microservice Architecture

Telecommunications carriers have recently come to adopt a microservice architecture in which small services operate in cooperation to form one service. Microservice architectures are widely used to speed up updates and scaling. In the microservice architecture, functions that were large components in a monolithic architecture are decomposed into small independent functions, thus the number of monitoring targets becomes larger than that in the conventional architecture.

The monitoring process uses resources such as network bandwidth for acquiring information from monitoring targets and disk capacity for accumulating the acquired information. In the microservice architecture, due to the enormous number of devices to be monitored, the amount of resources used in the monitoring process (overhead) increases [9]. Therefore, to reduce the overhead, the monitoring process is required to reduce the query rate as much as possible when the service is in a normal state.

C. Monitoring System

There are two kinds of monitoring implementation methods: agentless monitoring and agent-based monitoring.

Agentless monitoring involves collecting information without installing software for monitoring on physical or virtual devices to be monitored. Examples include monitoring the life and death of a server by pings or checking the availability of the service by transmission control protocol (TCP) port monitoring. However, for service assurance, it is necessary not only to simply monitor life and death but also to acquire detailed state transitions of service resources. Thus, further agent-based monitoring is required.

Agent-based monitoring requires two components: agents and the manager. An outline of the agent-based monitoring system is shown in Fig. 1. Agents are software for data gathering from monitoring target devices (e.g., physical servers, virtual servers, docker containers). The manager is software for data aggregation. When the manager requests information from each agent, agents transmit the latest data of a service resource to the manager. Zabbix [1] and Prometheus [2], which are open source software for agent-based monitoring, are widely used for service monitoring.

In the current agent-based monitoring system, the query rate is fixed for each target device to be monitored. For example, in Zabbix [1], the query rate setting exists under the name "update interval," and this must be entered as a fixed value. Although the query rate can be changed flexibly during the day and night, there is no function to change it dynamically at any time.

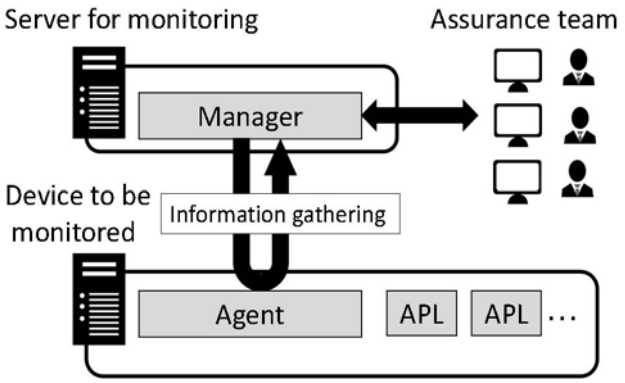

Fig.1 Outline of Agent-based Monitoring System

D. Related Work

For dynamically adjusting the monitoring interval, two main methods have been proposed: a threshold-based approach [5] and a prediction-based method [4, 6].

The threshold-based approach [5] modifies the monitoring interval on the basis of the absolute value of the difference between two consecutive metric values. For example, two kinds of threshold values a and b are prepared. When the difference exceeds the threshold value a, the monitoring interval decreases at a certain rate, and when the difference is less than the threshold value b, the monitoring interval increases at a certain rate.

The prediction-based approach [6] modifies the monitoring interval on the basis of the history of the most recent metric measurements. Specifically, the predicted value of the next metric x_p is calculated on the basis of the latest n metric history $x_1, \ldots, x_n$. When the next actual measured value x_{n+1} arrives, the differences between the actual two consecutive metric values $x_{n+1} - x_n$ and the prediction error $x_p - x_n$ are compared.

SAM [4] proposed by Tangari et al. is classified as a prediction-based method. SAM adopts AIMD for adjusting a window size used to calculate a predicted value of the metric x_p. If it is decided to extend the monitoring interval from the immediately preceding interval, the window size N used to calculate the predicted value is increased by one, and in other cases, N is halved. AIMD can discard the history of metrics that have become obsolete due to drastic changes in the metric tendency. SAM has achieved better monitoring accuracy (evaluated in terms of Root Mean Square Error (RMSE)) with less monitoring frequency than the previous methods [5, 6] in many test cases.

However, SAM tends to respond to small and constant variations in the amount of change and to frequently reduce the monitoring interval. In service resource monitoring, resources remain healthy for many hours, otherwise the service becomes unusable with frequent maintenance time. Therefore, SAM does not effectively reduce monitoring overhead in service resource monitoring. Thus, a new adaptive monitoring rate control method is required that is resistant to constant noise.

III. PROPOSED METHOD

This section proposes a new monitoring rate control method to solve the problems described in subsection II-D.

The proposed method assumes the use of agent-based monitoring software. The proposed method first removes noise in agents and then adjusts the monitoring interval on the basis of the prediction of the metric in the manager.

First, we first describe the overall algorithm of the proposed method. Then, the outline of the noise elimination of the metrics in the agent is explained, and finally, the algorithm of the prediction-based monitoring rate control method in the manager is described.

A. Outline of the Proposed Method

The algorithm of the whole proposed method is as follows.

- Agents acquire the time series data from each device to be monitored and store the information in the data storage part of the agent. Old data other than the latest period N is discarded at any time.
- The agent removes noise by referring to the most recent data in period N, creates new period N data, and stores it in the data storage unit of the agent.
- The manager requests the agent to send the latest data at a specific interval. At this time, both data before noise removal and data after noise removal are requested.
- When the manager receives the latest data from the agent, the manager stores both the data before noise removal and the data after noise removal in the data storage part of the manager. The data before noise removal is used for presentation to operators, and the data after noise removal is used for analysis for adjusting the monitoring rate. After the data is stored, the timing of requesting the agent to send information is determined by the monitoring rate control method.

B. Monitoring Rate Control Algorithm in the Manager

In this method, the degree of metric changes is calculated in order to determine the degree of adjustment of the monitoring rate. In the method of comparing the time series patterns of a fixed window size and defining the abnormality degree (e.g. Singular Spectrum Transformation), a delay occurs between the change of the metric and the increase, of the abnormality degree. Since the monitoring rate adjustment needs to be quick in response to metric changes, delay in detecting changes is not preferable. The change finder [10] can react quickly to changes in metrics, but the range of abnormality values varies greatly depending on the parameters. Since the abnormality value is used to calculate a new monitoring rate, the abnormality value need to be within a certain range. The manager executes the monitoring interval adjustment method on the basis of the cosine similarity. This method can detect changes in metrics without delay and handle the degree of abnormality within a certain range.

The monitoring interval adjustment method described in this section consists of four steps: calculate prediction (I), calculate change score on the basis of vector similarity (II), fit change score set by normal distribution (III), and determine monitoring interval (IV). The algorithm is described in detail below.

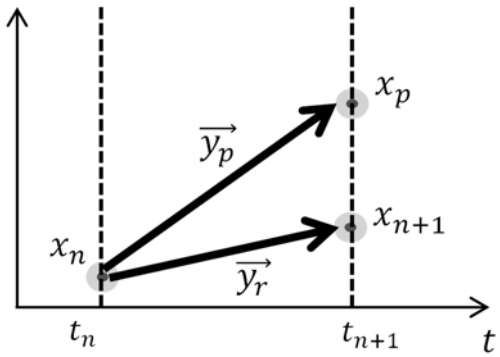

Fig.2 Prediction Vector and Actual Measured Value Vector

I. For the noise-removed data sequence $x_o, x_1, \dots, x_n$ and the time stamps $t_o, t_1, \dots, t_n$ corresponding to each data point, the predicted value x_p at time t is calculated with formula (1), where N is the window size.

$$x_p = x_n + \frac{t_{n+1} - t_n}{N-1} \sum_{i=n-N+1}^{n-1} \frac{x_{i+1} - x_i}{t_{i+1} - t_i} \tag{1}$$

II. Let the actual measured value at time t_{n+1} be x_{n+1}. By defining the prediction vector $\overrightarrow{y_p}$ and the actual measured value vector $\overrightarrow{y_r}$ as shown in Fig. 2, the cosine similarity $S(\overrightarrow{y_p}, \overrightarrow{y_r})$ can be calculated with formula (2).

$$S(\overrightarrow{y_p}, \overrightarrow{y_r}) = \frac{\langle \overrightarrow{y_p}, \overrightarrow{y_r} \rangle}{\|\overrightarrow{y_p}\| \cdot \|\overrightarrow{y_r}\|} \tag{2}$$

The change score $a(x_{n+1})$ of data x_{n+1} at time t_{n+1} is defined as follows.

$$\text{if } S(\overrightarrow{y_p}, \overrightarrow{y_r}) \geq 0 :$$

$$a(x_{n+1}) = 1 - S(\overrightarrow{y_p}, \overrightarrow{y_r}) \tag{3}$$

$$\text{else :}$$

$$a(x_{n+1}) = 1 \tag{4}$$

III. Assuming that the data point set obtained by logarithmically transforming the change scores $a(x_{n-N+1}), \dots, a(x_n)$ corresponding to the data points $x_{n-N+1}, \dots, x_n$ follows a normal distribution, the parameters of the probability density function of the normal distribution are estimated.

IV. For the probability density function obtained in step III, the α_1% points and α_2% points of the probability distribution are taken as α_1 and α_2, respectively, and the interval T_{new} to the next data point is determined by formulas (5), (6), and (7), where T_{max} and T_{min} are the upper and lower limits of the monitoring interval, respectively.

$$\text{if } a(x_{n+1}) \geq \alpha_1 :$$

$$T_{new} := T_{max} \tag{5}$$

$$\text{if } \alpha_1 \le a(x_{n+1}) \le \alpha_2 :$$
$$T_{new} \coloneqq \frac{T_{max} - T_{min}}{\alpha_1 - \alpha_2}(a(x_{n+1}) - \alpha_1) + T_{max} \quad (6)$$

$$\text{if } a(x_{n+1}) \le \alpha_2 :$$
$$T_{new} \coloneqq T_{min} \quad (7)$$

With the above procedure, T_{new} can be determined at any time. Statistical analysis is performed on the degree of abnormality in the history of metric changes, and the monitoring interval is reduced when the degree of abnormality is determined to be sufficiently large, so the proposed method is less susceptible to weak fluctuations.

IV. Generation of Time Series Data

In this section, time series data is created to execute the proposed method and the previous method.

To execute the algorithm, we emulate time series data as service resource data. The emulated time series data contains abnormal fluctuations assuming a service failure. We created multiple test cases with different types of long-term trends and different anomaly arrival rates.

A. Emulate Time Series Data

The time series data is created by synthesizing three time series: base data representing stationary noise, trend data representing long-term fluctuation tendency, and abnormal data representing sudden data fluctuation assuming a service failure. The number of data points is 10,000, and the time stamp interval is 5 seconds.

The base data is created by using a random function on the basis of the Poisson arrival process. Fig. 3 shows an example of the base data. It represents the constant noise that exists in CPU usage, memory usage, network traffic, etc.

Three types of trend data are prepared: no trend, upward trend, and periodic fluctuation. The degree of use of the service varies depending on the number of days that have elapsed since the service was started and the time zone. Many types of service resource data are affected by such a trend change. As an example, Fig. 4 shows the outline of the trend data of an upward trend case.

Two types of anomaly data are prepared: low-frequency anomaly and high-frequency anomaly. The burst of monitored metric is assumed to be an anomaly event that appears in service resource data. An anomaly event occurs every 5 seconds at 0.1% in the low-frequency case and 0.5% in the high-frequency case. Fig. 5 shows an example of the anomaly data in the low-frequency case.

The simulated service resource data used in the experiment is obtained by merging base data, trend data, and abnormal data. As an example, the experimental time series data in the case of the upward trend and the low-frequency burst are obtained as shown in Fig. 6 by combining the data in Fig. 3, 4, and 5.

B. Normal Phase and Anomaly Phase

In simulated time series data, normal phases and anomaly phases can be defined.

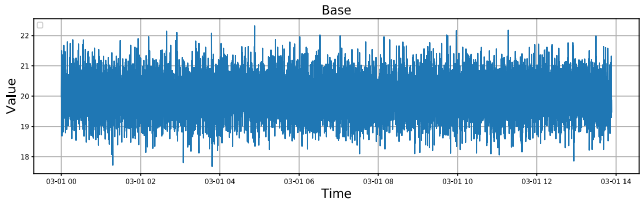

Fig.3 Example of Base Data

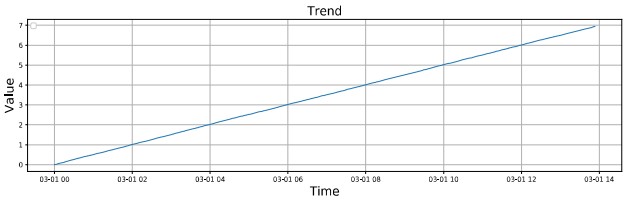

Fig.4 Example of Trend Data (Upward Case)

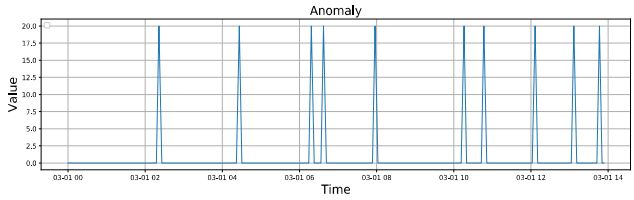

Fig.5 Example of Anomaly Data (Low-frequency Case)

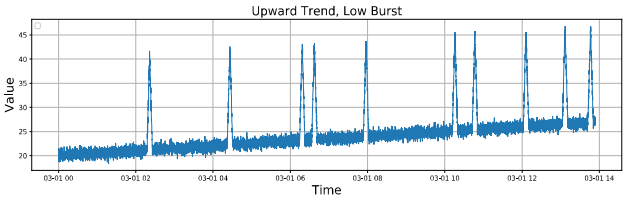

Fig.6 Time Series Data after being Synthesized (Upward Trend and Low-frequency Burst Case)

The normal phase is the part where no anomaly occurs in the service and only the stationary noise and trend appear in the time series data. Considering that the simulated time series data can be decomposed into three components of base, trend and anomaly, the value of the anomaly component in the normal phase is 0.

The anomaly phase is the part where the value sharply rises and steeply falls in the simulated time series data. In other words, in the anomaly phase, the anomaly component in the time series data is larger than 0.

The monitoring interval adjustment method in the service resource monitoring is required to keep the monitoring

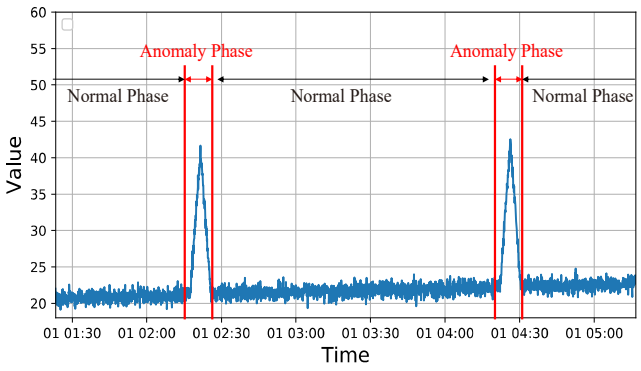

Fig.7 Normal and Anomaly Phases in Time Series Data

interval long and the overhead low in normal phases and to shorten the monitoring interval in anomaly phases to obtain precise information. In short, different behaviors are required in the normal and anomaly phases. Therefore, in this paper, we evaluate the performance of the proposed method and the prior method in normal and anomaly phases.

V. Evaluation

This section evaluates the performance of the proposed method and SAM. The proposed method and SAM were applied to the created time series data, and experiments were performed while changing the parameter (max monitoring interval). We evaluate the average monitoring intervals for anomaly and normal parts in time series data.

A. Experiment

Since there are three types of trends (no trend, upward trend, and periodicity trend) and two types of anomaly occurrence rates (low-frequency anomaly and high-frequency anomaly), the proposed method and SAM were implemented in six test cases..

- No Trend and Low-frequency Anomaly
- No Trend and High-frequency Anomaly
- Upward Trend and Low-frequency Anomaly
- Upward Trend and High-frequency Anomaly
- Periodicity Trend and Low-frequency Anomaly
- Periodicity Trend and High-frequency Anomaly

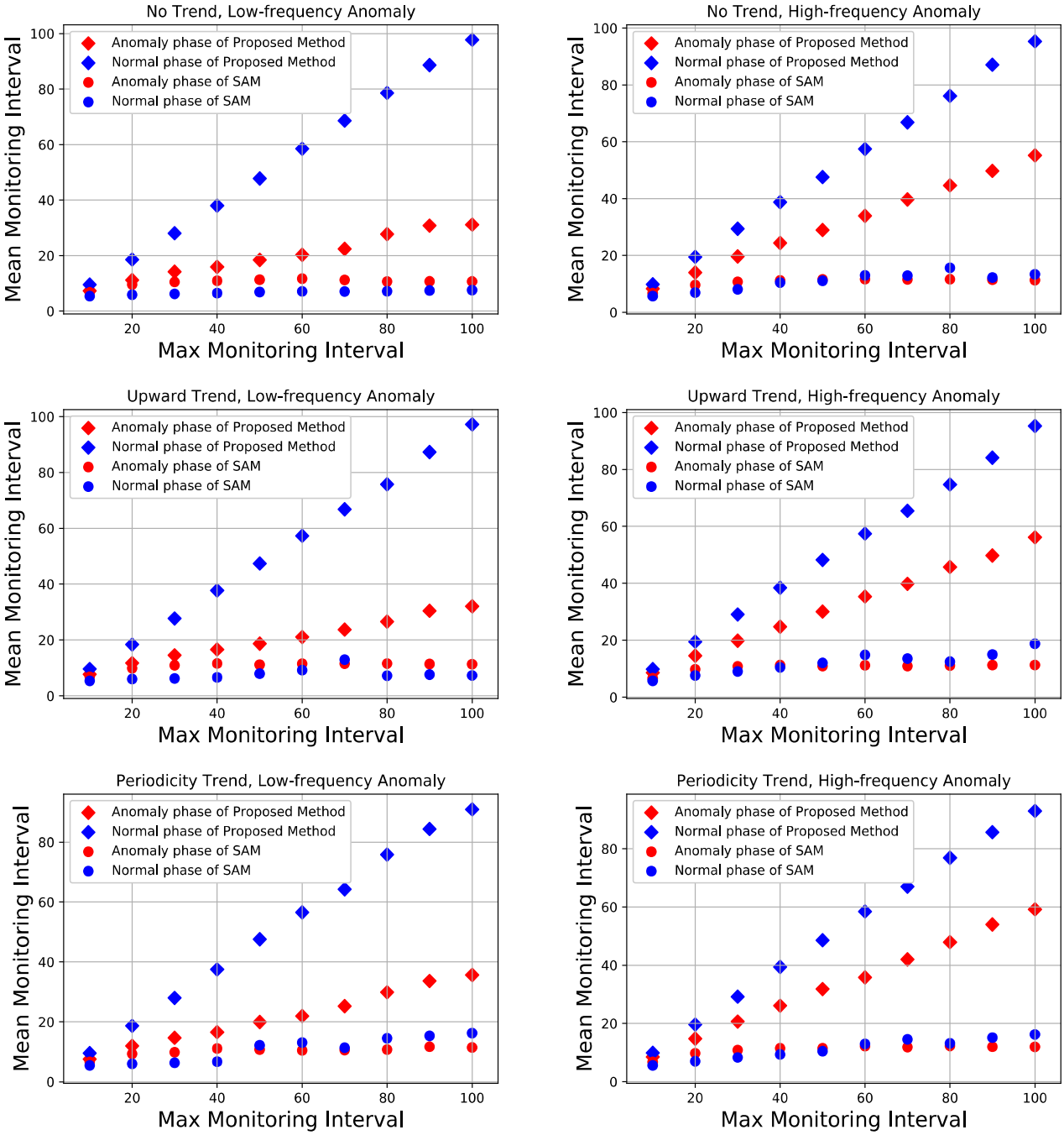

Fig.8 Experimental Results of Proposed Method and SAM

The minimum monitoring interval was set to 5 seconds, because the time stamp interval of the created teat case data is 5 seconds. The maximum monitoring interval was 10 to 100 seconds, and the experiment was performed with a shift of 10 seconds (10 patterns in total). The parameters in the proposed method α_1 and $\alpha_2$were set to 0.6 and 0.95, respectively.

Both the proposed method and SAM require past data to adjust the monitoring interval. Therefore, the first 1000 data of the test case data were used as pre-learning data.

Fig. 8 shows the results of applying the proposed method and SAM to six patterns of test case data and summing up mean monitoring intervals for normal and anomaly phases.

B. *Discussion*

In all test cases, the proposed method keeps the monitoring interval in the normal phase close to the maximum interval. In addition, it shortens the monitoring interval by the normal phase in the anomaly phase in all cases. The proposed method can adjust the monitoring interval in response to an abnormal sudden increase in the value, with almost no effect from the normal trend and stationary noise.

In all cases, results of SAM show almost no difference in the average monitoring intervals between normal and anomaly phases. In addition, in both phases, SAM selects a monitoring interval that is shorter than the interval in the anomaly phase of the proposed method. The reason for this is that SAM reacts to stationary noise and constantly tries to reduce the monitoring interval.

Although SAM is superior to the proposed method in terms of accuracy of monitoring in anomaly phases, SAM cannot be expected to reduce overhead in normal phases. The proposed method can reduce the overhead more effectively than SAM in normal phases. In addition, the proposed method can detect the essential fluctuation and adjust it dynamically for precise monitoring. The proposed method provides both accuracy in the event of service failure and overhead reduction in normal times, which are required for service resource monitoring.

Comparing the results of the proposed method for the patterns of low- and high-frequency anomalies, the performance of shortening the monitoring interval in the anomaly phases deteriorates more in the high-frequency anomaly case than in the low-frequency anomaly one. This is because in the high-frequency anomaly case, the anomaly phase arrives in a very short span. When the anomaly phase continues without a long normal phase, the proposed method learns the change degree of the metric in the anomaly phase and has difficulty reacting to the rapid increase or decrease of the metric. Therefore, the proposed method is more suitable for monitoring data with a low frequency of service resource anomalies.

Comparing the results of the proposed method with the patterns of periodicity trend and other trends, the effect of narrowing the monitoring interval is slightly weaker in the periodicity trend than in the other trends. By adding a periodicity trend, the range of fluctuation pattern in the value during normal phases expands. It is considered that the proposed method had difficulty reacting to the fluctuations in anomaly phases because it had learned such various fluctuation patterns. There is almost no difference in the results of the proposed method for the patterns of upward trend and no trend. The periodic trend given in this paper is about a 4-hour cycle. However, the periodic trends in the services of telecommunications carriers are brought about by people's behaviors, so most of them are daily or monthly cycles. Therefore, the periodicity trend existing in the actual service can be approximated to the upward trend or the downward trend, and its effect is considered to be negligible.

VI. Conclusion

In this paper, we proposed a novel adaptive monitoring rate control method for service assurance that is robust to constant noise in service resource metrics. Our method uses cosine similarity to measure the change degree of value, so the monitoring interval can be shortened only when an essential change occurs.

Evaluations conducted using various test cases show that the proposed method can reduce the monitoring overhead more effectively than the previous method and can also ensure the accuracy of monitoring when a service abnormality occurs. However, the learning accuracy deteriorates when a service abnormality frequently occurs because the proposed method learns the abnormality.

In future work, we will attempt to improve the performance when an anomaly frequently occurs by adding a dynamic window size adjustment function to the proposed method. In addition, although the monitoring interval is adjusted for a single metric in this paper, we are considering expanding the method to support multiple metrics in the future.

References

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

[2] Prometheus, https://prometheus.io/

[3] G. Yang, K. Wang, and X. Zhou, "An adaptive resource monitoring method for distributed heterogeneous computing environment", Parallel and Distributed Processing with Applications, 2009 IEEE International Symposium on, pages 40–44. IEEE, 2009.

[4] G. Tangari, D. Tuncer, M. Charalambides et al., "Self-Adaptive Decentralized Monitoring in Software-Defined Networks", IEEE Transactions on Network and Service Management, 2018, pp. 1277-1291.

[5] S. R. Chowdhury, M. F. Bari, R. Ahmed, and R. Boutaba, "PayLess: A Low Cost Network Monitoring Framework for Software Defined Networks," in Proc. IEEE/IFIP NOMS, Krakow, Poland, May 2014, pp. 1–9.

[6] T. Zhang, "An Adaptive Flow Counting Method for Anomaly Detection in SDN," in Proc. ACM CoNEXT, Santa Barbara, CA, USA, Dec. 2013, pp. 25–30.

[7] Z. Fu and N. Venkatasubramanian, "Adaptive Parameter Collection in Dynamic Distributed Environments", The 21st International Conference on Distributed Computing Systems, 2001, pp. 469-478.

[8] B. Adipat and D. Zhang, "A real-time adaptive traffic monitoring approach for multimedia content delivery in wireless environment", Systems Man and Cyvernetics 2003 IEEE International Conference, 2003, pp. 280-285.

[9] P. Las-Casas, J. Mace, D. Guedes, and R.Fonseca, "Weighted Sampling of Execution Traces: Capturing More Needles and Less Hay", SoCC '18, October 11–13, 2018, Carlsbad, CA, USA, pp326-332.

[10] J. Takeuchi and K. Ymanishi, "A Unifying Framework for Detecting Outliers and Change Points from Time Series", IEEE Transactions on Knowledge and Data Engineering, Vol. 18, No. 4, 2006, pp. 482-49