Research on Microservice Application Performance Monitoring Framework and Elastic Scaling Mode

Zhihui WANG¹, Ying XIA², Changhua SUN³, Lei CHENG¹,*

¹Information and Communications Testing and Simulation Center, China Electric Power Research Institute, Beijing, China
²Safety and Quality Department, State Grid Information & Telecommunication Group Co., LTD., Beijing, China
³Power Technology Center, State Grid Zhejiang Electric Power Co., LTD. Research Institute, Zhejiang, China

*hesoyam123456@sina.com

Abstract—Compared with the traditional monolithic architecture, the microservice architecture style divides a system into different microservices which run in the distributed system. The complex dependencies between services bring new challenges to the monitoring analysis and quality assurance of system performance. According to the characteristics of microservice application, a performance monitoring framework based on big data is designed in this paper. It monitors and controls the microservice performance through data collection, big data storage, elastic scaling management, integrated scheduling and so on. Furthermore, an elastic scaling mode based on time sliding window and scene driven is proposed. Experiments show that this mode could realize resource expansion prediction and resources saving. This research is helpful to real-time monitoring and continuous optimization for microservices, which will effectively promote the integration process of development, testing and maintenance for microservice application in SGCC (State Grid Corporation of China).

1. INTRODUCTION
The microservice style is an architecture that divides a traditional software system into different modules in the form of microservices. Each service focuses on a single business, runs in a separate process and may have its own database. They apply lightweight communication mechanism to communication with each other. Due to the low coupling splitting, distributed architecture and communication mechanism of microservice applications, higher requirements are put forward for its performance monitoring [1]. Normally in a microservice application, users need to wait for several microservice sequential calls so as to complete a business operation. That means the output data of one microservice will be the input data of the next microservice. The dependency between services may affect the performance of the whole business call. In order to effectively guarantee the good operation of servers and provide reliable information services, enterprises should have its server performance monitoring management [2]. Therefore, performance real-time monitoring, effective analysis and intervention in time are the key to ensure efficient operation of the software. In this paper, a microservice application performance monitoring framework is proposed. It realizes the real-time monitoring and microservice performance analysis through data collection, data storage, elastic scaling...
management, integrated scheduling and other modules. In addition, aiming at the elastic scaling of cloud services in the framework, an elastic scaling mode based on scene driven and time sliding window technology is presented through the performance indicators calculation and prediction. Experiments prove that this mode could effectively predict resource scaling and save resources. This framework is conducive to real-time monitoring and continuous optimization for microservice applications, which promotes the integration process of research, development, testing, operation and maintenance for microservices in SGCC (State Grid Corporation of China).

2. PERFORMANCE MONITORING ANALYSIS

2.1 Monitoring Objects
Performance monitoring is used to record the statistical results of performance data in a period of time, compare the results at several time points and find the abnormal point or some rules [3]. Server performance evaluation indicators include response time, response delay, system throughput, resource utilization, system bandwidth and so on [4]. In this paper, CPU (Central Processing Unit) utilization and response time are used as the main objects of performance monitoring. In the process of performance monitoring, the performance data obtained at a certain time may be a certain accident, which cannot fully reflect the system performance in the current period. Therefore, the historical data should be analyzed firstly and then the normal range and historical average value of the data can be obtained. Next, compared with the current value, more reasonable indicator value is calculated. In this way, more accurate basic data are provided for performance analysis [5]. Time sliding window is a common data flow model. It selects part of the data and sets the starting and ending positions of the window. The start and end positions will change with the new data. The direction of change is moving towards the arrival of new data, so that the old data will be removed from the window and the new data will be added to the window [6]. Time sliding window technology is very suitable for the acquisition and analysis of performance indicators in a short time, thus the performance indicators are smoothed and predicted based on this technology in this paper.

2.2 Elastic Scaling
Elastic scaling refers to the mechanism that the cloud data center automatically matches the amount of resources according to the change of load demand and guarantees the service ability of the cloud data center business system continuously and adaptively without excessive waste of resources [7]. Under the condition of mastering the real-time performance data of microservices, dynamic real-time adjustment of microservice resources is the premise to ensure their reliable operation. For the operation mechanism of elastic scaling, the most important thing is to solve the problem of starting time which mainly includes two methods: threshold warning and prediction starting. In reference [8] [9], the threshold warning is adopted. A threshold for CPU usage should be preset and the elastic scaling is started when the actual indicator exceeds the threshold. Reference [10] [11] adopts the prediction starting which establishes data trend model through historical data firstly, then uses current data to predict the resource demand at the next moment, and finally decides whether to start the elastic scaling. The advantage of threshold warning is simple and intuitive with less calculation. The advantage of predictive starting is that it can predict the performance indicators in advance, rather than deal with the performance bottlenecks. In addition, it is worth noting that the cooling time of elastic scaling refers to a period of locking time after a scaling activity is successfully executed, which means the scaling group does not perform other scaling activities. Therefore, in this paper the performance monitoring time is divided into two types based on the idea of scene driven in order to maximize the use of microservice resources. They are business peak and business low peak which use two starting time methods respectively.

3. PERFORMANCE MONITORING FRAMEWORK
The module composition and working mechanism of microservice performance monitoring framework are shown in Fig. 1. First of all, the performance testing tool starts testing the performance by executing
test scripts. After that, the response time and resource indicators are obtained through the data collector and then stored based on big data. Finally, the automatic elastic scaling mode runs according to the calculation of performance indicators and related algorithms. Integrated scheduling tool is used to initiate and schedule tasks for performance monitoring.

Figure 1. Microservice performance monitoring framework

3.1 Integrated Scheduling Tool
As the most important part of this framework, integrated scheduling tool runs through the whole process of performance monitoring. It does not carry out the actual business work, but connects the various parts of performance monitoring, so that it is responsible for the distribution and scheduling of tasks. Its advantage is that it can effectively organize all parts of performance monitoring, but also effectively reduces coupling. Therefore, replacing any part will not affect the operation of other parts in the framework. Jenkins, an open source continuous integration tool, can be used as a typical integrated scheduler.

3.2 Data Collector
The data collector mainly obtains response time and resource indicators. The response time is collected by the feedback of each business scenario node in the performance testing tool, and the resource indicators are mainly collected by the performance agents embedded in each server. The response time is obtained by the performance test tool when it runs the scripts, and is transferred to data collector. Resource indicators are collected by the data collector directly through the agents. Finally, these two kinds of data are summarized and prepared for data storage.

3.3 Big Data Storage
Performance monitoring has the following characteristics: long monitoring time, many kinds of indicators and multidimensional data analysis requirements, which are very suitable for big data storage. This part will create different time window data, compare the performance data in the current time window vertically and horizontally, and then analyze the data. After the current time window data is stored, the performance data is further processed. Finally, the basic data required by the elastic scaling algorithm are transmitted to the elastic scaling manager. In addition, big data storage provides a decision support platform, which is used to display the multi-dimensional performance indicators, related reports and charts of the year, quarter and month.
3.4 Elastic Scaling Manager
At the end of a time window, the elastic scaling manager obtains the required basic data, smooths the
time window data, and applies these data to the elastic scaling algorithm. Then, the microservice
resources are adjusted by matching different configured thresholds so as to improve resource utilization.
The elastic scaling mode is explained in detail in next chapter.

3.5 Others
Performance monitoring personnel could view the real-time performance indicators of each link of
microservices in the decision support platform and manually allocate the microservice resources
through this management tool. In some specific scenarios, manual intervention is able to respond to the
needs of scenarios accurately and quickly, avoiding the low performance of microservices.

4. ELASTIC SCALING MODE
According to the business characteristics of SGCC, in this paper, the elastic scaling mode is divided
into two stages based on the business peak and the business low peak. In the peak business hours, the
resource requirements are high. In order to avoid the performance problems caused by large amount of
parallel access, it is necessary to predict the resource utilization in the next moment as early as possible,
so the prediction starting method is adopted. In the low peak business period, the cost and efficiency of
resource utilization are pursued while satisfying business access, so the threshold warning method is
adopted.

4.1 Elastic Scaling Algorithm
For the CPU utilization, the current monitored data is compared to the threshold directly and then
elastic scaling is started when the threshold is exceeded. This method is accidental and cannot fully
reflect the current use of resources in a period of time. For example, the current running state does not
require elastic scaling, but because the instantaneous value exceeds the threshold value, the expansion
of the resource is started. Also, it needs a certain cooling time, which causes waste of resources to a
certain extent. Therefore, it is necessary to ensure the smoothness of the data. Time sliding window
technology can divide the monitored performance data into windows and form a window set as \( W = \{ w_1, w_2, \ldots, w_n \} \). Each window \( w_t \) corresponds to a period of time. The performance data set of the indicator
and the number of requests can be obtained, so that the average performance data \( \text{Avg}(w_t) \) of this
window can be calculated. The calculation method for the window performance data \( \text{Win}(w_t) \) for the
current window is to obtain the performance data of the previous window at the end of the current
window, multiply the attenuation factor \( r (r = 0.1) \) and then add the average performance data of the
current window, namely,

\[
\text{Win}(w_t) = \text{Win}(w_{t-1}) \times r + \text{Avg}(w_t).
\]  

(1)

In the low peak period, the performance data is smoothed based on the time sliding window and then
it is compared with the threshold \( T_L \) of the low peak period. When the threshold is exceeded, elastic
scaling is started. In the business peak period, based on the smoothed historical data, the performance
data of the next window is predicted. The predicted performance data is compared with the peak time
threshold \( T_H \). If it is greater than the threshold, elastic scaling is started. The prediction method is
autoregressive moving average model in which the parameters \( \alpha \) and \( \beta \) are 0.85 and 0.15, namely,

\[
\text{Win}(w_{t+1}) = \alpha \times \text{Win}(w_t) + \beta \times \text{Win}(w_{t-1}) + (1-(\alpha+\beta)) \times \text{Win}(w_{t-2})
\]  

(2)

In conclusion, the elastic scaling algorithm of the microservice performance monitoring framework
is shown in Fig. 2.
4.2 Experiments

The experiments of elastic scaling mode is carried out by simulating real business scenarios based on the microservice application automation testing framework [12] and the automatic test data generation method [13] which are proposed by the author's research team.

During the business low peak, five periods are selected for the experiments. As the elastic scaling mode described above, performance indicators are smoothed compared with those not smoothed, and the start-up time of elastic scaling is controlled by the same threshold \( T_L \). The results are shown in TABLE I. After smoothing the performance indicators, the count of elastic scaling decreased by 6.7% to 27.3% when the business response time increased only by 2.9% to 6.9%. In the low peak period, it is feasible to increase a small amount of business response time in exchange for some running resources, which would save the server resources and cost to a certain extent.

![Figure 2. Elastic Scaling Algorithm](image-url)

**Input:** History window performance data queue \( Q \)  
**Output:** Updated window performance data queue \( Q \)  

**Algorithm:**

//Get all the request data of the current window \( w_t \), and calculate the average CPU utilization of the \( w_t \)

\[
\text{totalCpuUsage} = Q.getTotalCpuUsageInWindow(w_t);
\]

\[
\text{requestCount} = Q.getRequestCountInWindow(w_t);
\]

\[
\text{Avg}(w_t) = \frac{\text{totalCpuUsage}}{\text{requestCount}};
\]

//Get the CPU utilization window data of the previous window

\[
\text{Win}(w_{t-1}) = Q.getAvgCpuUsageInWindow(w_{t-1});
\]

//Calculate the CPU utilization window data of the current window according to formula (1)

\[
\text{Win}(w_t) = \text{Win}(w_{t-1}) \cdot r + \text{Avg}(w_t)
\]

//Update window performance data queue \( Q \)

\[
Q.setAvgCpuUsageInWindow(w_t) = \text{Win}(w_t)
\]

if ( dateTime.NOW In business low peak period){
  if ( \( \text{Win}(w_t) > T_L \) ){
    StartElasticScaling();
  }
}

else if ( dateTime.NOW In business peak period){
  //Get the CPU utilization window data of the last two windows

  \[
  \text{Win}(w_{t-2}) = Q.getAvgCpuUsageInWindow(w_{t-2});
  \]

  //Calculate the CPU utilization window data of the next window according to formula (2)

  \[
  \text{Win}(w_{t+1}) = \alpha \cdot \text{Win}(w_t) + \beta \cdot \text{Win}(w_{t-1}) + (1-(\alpha+\beta)) \cdot \text{Win}(w_{t-2})
  \]

  if ( \( \text{Win}(w_t) > T_L \) ){
    StartElasticScaling();
  }
}
During the business peak, five periods are selected for the experiments and each period containing 30 time windows. The method introduced in this paper is used to predict the performance indicators, and the prediction effect is shown in TABLE II. The results show that the average error of CPU utilization is between 9 and 14 and the minimum error is 0. The prediction method could effectively predict the performance indicators to a certain extent and prepare for elastic scaling in advance, which would avoid dealing with resource bottlenecks due to the sharp increase of business volume during peak period.

### TABLE II. CPU UTILIZATION PREDICTION IN LOW PEAK PERIOD

| No. | Number of windows | Minimum error | Maximum error | Average error |
|-----|-------------------|---------------|---------------|---------------|
| 1   | 30                | 1             | 21            | 13            |
| 2   | 30                | 0             | 15            | 12            |
| 3   | 30                | 3             | 12            | 9             |
| 4   | 30                | 1             | 19            | 14            |
| 5   | 30                | 2             | 16            | 11            |

5. CONCLUSION
A microservice application performance monitoring framework is proposed in this paper, which monitors and controls the microservice performance through data collection, big data storage, elastic scaling management, integrated scheduling and so on. Furthermore, an elastic scaling mode is presented based on scene driven and time sliding window technology. Experimental results show that this elastic scaling mode could effectively predict resource scaling and save resources. In the future, combined with the relevant standards in SGCC, a full link monitoring tool based on cloud environment can be developed based on this monitoring framework. The effects of reducing service response time or reducing embedded agents on the performance of microservices will be further researched.

ACKNOWLEDGMENT
This research was supported by State Grid Corporation of China Science and Technology Project (5700-201941223A-0-0-00). We would like to acknowledge our sponsors for their financial support, and as well as thank the reviewers for their valuable feedback.

REFERENCES
[1] X. W. Wang and L. Du, “State Grid Corporation of China Science and Technology Project Feasibility Study Report on Testing System and Key Technologies for Microservice Applications,” unpublished.
[2] N. Guo and X. L. Zheng, Software Engineering Practical Tutorial, 2rd ed., Post & Telecom Press, 2010. pp. 5-6.
[3] Y. H. Xie, “Design and Implementation of Information Platform Server Performance Monitoring System,” Diss. University of Electronic Science and Technology of China, Jun 2016.
[4] H. Chen, “Research on Server Performance Analysis and Prediction Based on SVM Algorithms,” Diss. Nanchang University, May 2019.

[5] N. Doñamayor, J. Dinani, M. Römisch, Y. Zheng and T. F. Münte, Performance Monitoring During Associative Learning and Its Relation to Obsessive-Compulsive Characteristics, Biological Psychology, vol. 102, no. 1, pp. 73-87, 2014.

[6] Z. J. Lv, “Sliding Window TOP-K Monitoring Over Distributed Data Streams,” Diss. Shandong University, Jun 2017.

[7] Z. Yang, “Research on Automatic Elastic Extension of Virtual Service Node Based on Scenario Driven,” Diss. Inner Mongolia University, May 2019.

[8] R. Mayer, B. Koldehofe and K. Rothermel, "Meeting Predictable Buffer Limits in the Parallel Execution of Event Processing Operators," in Proceedings of the 2nd IEEE International Conference on Big Data, pp. 402-411, 2014.

[9] T. Heinze, Z. Jerzak, G. Hackenbroich and et al, "Latency-Aware Elastic Scaling for Distributed Data Stream Processing Systems," in Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems, pp.13-22, 2014.

[10] L. N. Li, X. H. Wei, X. Li and X. W. Wang, Burstiness-Aware Elastic Resource Allocation in Stream Data Processing, Chinese Journal of Computers, vol. 41, no. 10, pp. 2193-2208, Oct 2018.

[11] N. Zacheilas, V. Kalogeraki, N. Zygoouras and et al, "Elastic Complex Event Processing Exploiting Prediction," in Proceedings of the IEEE International Conference on Big Data, pp. 213-222, 2015.

[12] Y. Wang, L. Cheng and X. Sun, “Design and Research of Microservice Application Automation Testing Framework,” in Proceedings of 2019 International Conference on Information Technology and Computer Application, pp. 257-260, 2019.

[13] H. X. Ding, L. Cheng and Q. Y. Li, “An Automatic Test Data Generation Method for Microservice Application,” in Proceedings of 2020 International Conference on Computer Engineering and Application, pp. 188-191, 2020.