IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 8 Issue 6, June 2021
ISSN (Online) 2348 – 7968 | Impact Factor (2020) – 6.72
www.ijiset.com

Examination of Auto Scaling Parameter Setting Method for Safety Information Systems

Takahiro Koita[1] and Yu Iguchi[1]

[1] Faculty of Science and Engineering, Doshisha University, Kyotanabe, Kyoto, 610-0321, Japan

Abstract

The purpose of this study is to examine setting methods for suitable conditions for Auto Scaling. Safety information systems are one measure taken when natural disasters such as earthquakes and tsunamis occur. However, since there is a large difference in how much the system is accessed normally as opposed to during the time of a disaster, it is difficult to manage its servers. By using the Auto Scaling function of Amazon EC2 provided by Amazon Web Service, it is possible to flexibly deal with the big gap in how many times the system is accessed. Auto Scaling is one of the functions offered by Amazon Web Services. By setting conditions for it, it automatically conducts scaling that increases or decreases servers. Because there are many conditions for Auto Scaling, it is difficult to ascertain which condition influences its operation. We clarify the influence of conditions' settings on Auto Scaling and its result through evaluation experimentation and examine the appropriate setting method.

Keywords: *Amazon Web Services, Auto Scaling, Safety Information Systems.*

1. Introduction

An armed attack or natural disasters such as earthquakes, tsunamis, typhoons, or volcano eruptions could occur at any time in our lives, and many measures are being taken to prepare for and counter these disasters. A safety information system is an information management measure for collecting and sharing the safety information of disaster victims in order to keep damage to a minimum. Safety information systems are most commonly operated using cloud services. However, as it is not possible to predict the timing of a disaster and the number of its victims, it is difficult to estimate the appropriate amount of resources necessary to effectively operate the system. A system that focuses only on either the response time performance or the cost performance cannot be deemed to be a satisfactory system for the demands of the users. Thus, it is necessary to offer a service without any excess or deficiency while maintaining the balance of the two performances.

Safety information systems collect and provide information on the situation of the disaster or the safety information of the victims when a disaster strikes. To be precise, the safety information includes the name, date of birth, gender, address, injury status, whether deceased or not, location, and contact information of a victim [1]. To summarize, an email with the URL of the website of the safety information system is sent to every registered user of the system. Disaster victims could register/check their safety information from this website. The cloud service is used to store this information. As the cloud service consists of data centers with high earthquake resistance, it can offer stable service during the time of disaster [2].

One of the characteristics of safety information systems is the difference in how much the system is accessed normally as opposed to during the time of disaster. Usually, it is rarely accessed, while during the time of disaster it is constantly accessed. Therefore, the system requires many servers to operate during the time of disaster. If it were operated during normal times with as many servers as required to process the number of people accessing it during the time of disaster, the number of servers would be excessive and the cost would be high. On the other hand, if it were always operated with the number of servers required to process the number of people accessing it during normal times, it may lead to a slow response time during the time of disaster.

The purpose of this study is to examine setting methods for suitable conditions for Auto Scaling. Auto Scaling is one of the functions offered by Amazon Web Services. By setting conditions for it, it automatically conducts scaling that increases or decreases servers. Because there are many conditions for Auto Scaling, it is difficult to ascertain which condition influences its operation. We clarify the influence of conditions' settings on Auto Scaling and its result through evaluation experimentation and examine the appropriate setting method.

2. Amazon Web Services

2.1 Amazon Elastic Compute Cloud

Amazon Elastic Compute Cloud (EC2) is a cloud service that offers a virtual server function used in safety information systems. Amazon EC is a service that allows the sharing of physical servers installed in the data centers of Amazon Web Services among the users of the service through virtual technology. As it can increase and decrease the number of the virtual servers called "instance," it could increase the processing capacity of the cloud in response to the load. The higher the number of instances, the higher the processing capacity of Amazon EC2.

2.2 Auto Scaling

One of the methods for adjusting the processing capacity of the cloud is Scaling, which increases or decreases the number of servers. Auto Scaling conducts this scaling automatically [3]. By increasing the instance number when there is an increase in load, such as a sudden upsurge in the number of accesses, and decreasing the instance number when the access number has calmed down and the load has gone down, it can freely expand/reduce the processing capacity.

In order to conduct Auto Scaling, it is necessary to set many conditions. As it is not clear which condition influences Auto Scaling and how, it is difficult to set suitable conditions. Each class of instance has an instance type with a different combination of CPU, memory, storage, network capacity, and hourly cost, which makes it possible to flexibly select an instance suitable for the purpose.

2.3 Instance Type

The definition of the specs of a server is called the "instance type." There are currently a few dozen instant types in Amazon EC2. In this study, t2.micro, which is a T2 instance, is used. Table 1 shows the capacity of t2.micro instance.

Table 1: Capacity of t2.micro Instance

vCPU	1
ECU	Changeable
Memory (GiB)	1
Instance Storage	Only EBS
Linux/UNI Price	$0.0152/Time
Baseline Capacity	10%
CPU Credit/Time	6

Linux/Unix price listed in Table 1 is the cost for running an instance for one hour. In this experiment, even if the time of usage was less than one hour, the cost is calculated as the cost of running the instance for one hour.

The reasons for using t2.micro instance include its low cost, as it is a general purpose instance, and the fact that it can burst. CPU usage has a standard called the "baseline." When that standard is exceeded, it starts to use a resource of CPU called "CPU credit." It enables the usage of the instance with higher capacity until the CPU credit is spent. When the accumulated CPU credit is spent, the capacity of CPU slowly returns to the baseline. The advantage of using burst is that it can keep the cost low when one does not need much computer resources normally but occasionally needs a larger amount of computer resources. As this function suits the characteristics of the safety information system, this study used the instances of the T2 series.

2.4 Issues in of Auto Scaling

One of the challenges facing the safety information system is to secure resources that are suitable for each situation. As there is a large difference in the number of people accessing safety information systems during normal times as opposed to a time of disaster, it leads to a higher cost. There is also the issue of delay in response time. If one focuses on keeping the cost down while scaling, one could operate the system with fewer instances. However, when the number of instances is small, the

processing capacity of the system decreases and it may lead to a delay in response time or even a system shutdown due to the high load. On the other hand, if one focuses only on shortening the response time, one could operate the system with many instances. However, the price of extra instances secured during the time of fewer accesses is an unnecessary cost because an increase in the number of instances leads to a higher cost. Thus, resources have to be secured flexibly according to the situation by using Auto Scaling. However, it is necessary to set the scaling conditions in order to conduct Auto Scaling. The factors of these conditions are called “parameters” in this study. As it is necessary to set many types of parameters in order to conduct Auto Scaling, there are also many combinations of these parameters. As the effective parameters for Auto Scaling for the safety information system have not been elucidated, it is difficult to set appropriate parameters.

3. Evaluation Experiment

3.1 Experiment Outline

In order to examine the appropriate parameter setting for Auto Scaling, we examined a setting that prioritizes the cost, a setting that prioritizes the response time, and a setting that balances the cost and the response time.

Many clients access the safety information system, and Amazon EC2 instance processes the accesses as the Web server. In this experiment, we did not create a database server. We generated the database processing time by repeating the same process many times using PHP as the writing or loading time of the virtual database. When the processing on the virtual database is complete, the Web server responds to the clients. By evaluating the response time and the number of instances, the parameters setting that could balance the cost and the response time is examined.

The parameters that displayed changes when a load was placed on an instance without Auto Scaling setting was as follows: “CPU usage,” “network input,” and “network output.” Thus, the settings of these three parameters were examined in this experiment.

3.2 Experiment Conditions

In order to elucidate the effect of the parameters setting on Auto Scaling, the same number of accesses and response time of the virtual database server were used. The number of accesses was set to increase/decrease with 1000 accesses as the peak. The processing of the virtual database is adjusted to have around 40(ms) response time when no load is placed on it.

As introduced in [4], the number of accesses to the safety information system reaches its peak in about 20 minutes after a disaster occurred, and the safety confirmation email was sent and starts to gradually decrease after that point. The setting for the number of accesses for this experiment reproduced this feature. As Amazon EC2 takes at least 5 minutes to start an instance, the load patterns were set in such a way as to reach the peak in 10 minutes and then gradually decrease in order to verify the increase/decrease of the instances. Figure 1 shows the graph of the thread number, which represents the number of accesses to JMeter, and the elapsed time. In addition, because the usage cost of Amazon EC2 is calculated with units of one hour, the one-hour usage cost of the instance is necessary each time an instance is started.

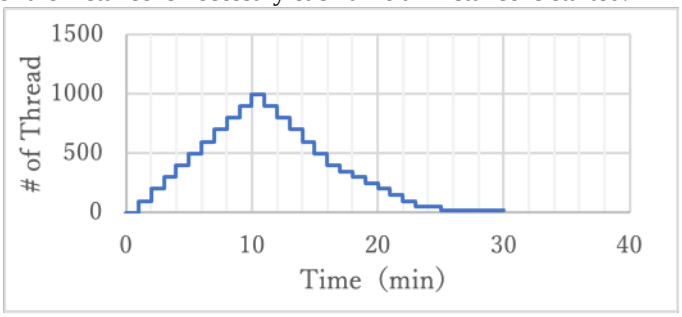

Figure 1 : Thread Number and Elapsed Time

From our previous study [5], the basic parameter range of CPU usage, network input, and network output were determined. The basic parameters are shown in Table 2. The parameter setting that prioritizes the cost and the parameter setting that prioritizes the response time are determined.

Table 2 : Basic Parameter Range

		Setting That Prioritizes Cost	Setting That Prioritizes Response Time
CPU Usage (%)	Upper Limit	80	10
	Lower Limit	30	N/A
Network Input (Byte)	Upper Limit	3500000	1000000
	Lower Limit	2000000	N/A
Network Output (Byte)	Upper Limit	2000000	1500000
	Lower Limit	1500000	N/A

3.3 Evaluation

In this experiment, a compromise was made between the setting that prioritizes the cost and the setting that prioritizes the response time, and the setting that could maintain the balance between cost and response time was explored while changing the parameters. The in-between setting is in discussed in [5]. In order to examine the best parameter setting, an evaluation experiment was conducted with several parameter patterns that use the in-between parameter as their standard. The parameter settings examined during this evaluation experiment are shown in Table 3. Setting A to C changed only the CPU usage. Setting A had a high threshold value, B had an average threshold value, and C had a low threshold value. Likewise, D to I changed only either the network input or network output and had three different threshold values in each group. Figure 2 shows the results of conducting Auto Scaling with these parameter settings.

Table 3 : Parameter Settings

	CPU Usage (%)		Network Input (Byte)		Network Output (Byte)	
	Lower Limit	Upper Limit	Lower Limit	Upper Limit	Lower Limit	Upper Limit
Setting A	50	80	1000000	2000000	700000	1700000
Setting B	60	90	1000000	2000000	7000000	1700000
Setting C	30	50	1000000	2000000	7000000	1700000
Setting D	10	50	2000000	3500000	700000	1700000
Setting E	10	50	1000000	2000000	700000	1700000
Setting F	10	50	1000000	3500000	700000	1700000
Setting G	10	50	1000000	2000000	1000000	2000000
Setting H	10	50	1000000	2000000	500000	1000000
Setting I	10	50	1000000	2000000	500000	2000000

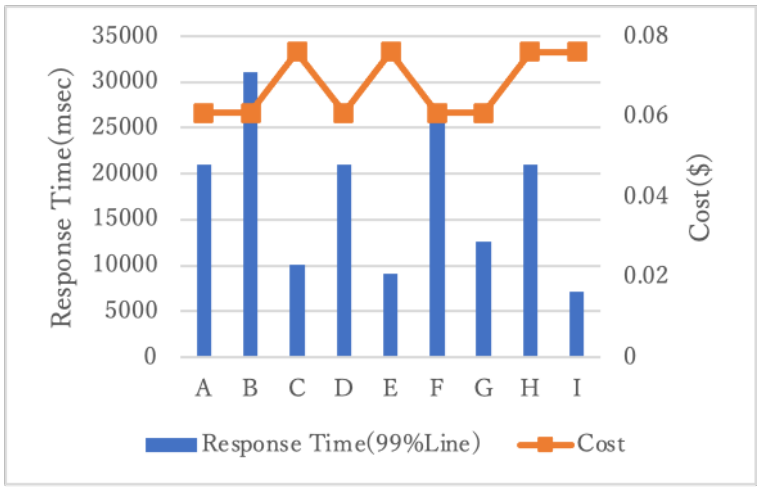

Figure 2 : Response Time and Cost for Each Parameter Setting

Depending on the parameter setting, there are cases such as Setting H and Setting I in Figure 2 where the response time is almost the same, but the cost is different. Among Setting A to I, the ones that have relatively shorter response times and lower costs are Setting A, Setting D, Setting G, and Setting I. The response time for Setting G stands out for being less than half that of Setting B.

3.4 Discussion

As shown in Figure 2, Setting G struck the best balance between cost and response time. It was the parameter setting that could offer a certain level of service while keeping the cost low. Setting G was a setting that changed only the network output of the in-between parameter setting.

The settings that had a short response time and low cost had higher threshold value than the in-between setting. Setting A had a higher CPU usage threshold value than the in-between setting, and Setting D had a higher network input threshold value. Though Setting I had the same high network output threshold value as Setting G, Setting G had a higher lower limit for scale-in. Moreover, settings though Setting A and Setting D had higher threshold values than the in-between setting (same as Setting G), and their response time did not shorten compared to Setting G. As Setting G was a setting with a short response time whose network output setting was changed, it is inferred that network output has the largest influence on Auto Scaling among CPU usage, network input, and network output.

4. Conclusion

This study conducted an evaluation experiment for the purpose of examining the parameter setting method that can maintain the balance of cost and response time in Auto Scaling used for safety information systems. In the evaluation experiment, several combinations of the following parameters, which were deemed to influence Auto Scaling, were set: CPU usage, network input, and “network output. Which parameter influences the performance of Auto Scaling was also examined.

Though only one type of instance was used for this experiment, processing with a combination of instances with different features must be considered because far more people accessing the system is expected during a real major disaster. Our future challenges are to thoroughly understand the capacities and characteristics of different instances and to consider an Auto Scaling method that can flexibly respond to unpredictable situations such as accessing safety information.

References

[1] Fire and Disaster Management, Agency of the Ministry of Internal Affairs and Communications, Safety Information System Outline (2013).

[2] Masaki Nagata, The Development of the scalable large-scale Safety Information system using AWS, IPSJ SIG Technical Report, 2013-CDS-Vol.8, No.10, pp.1-8 (2013).
[3] Amazon Web Services, https://aws.amazon.com/jp/.
[4] Masaki Nagata, Study on Turning Safety Ascertainment System into Global Redundant System and Access Prediction, Doctoral Thesis, Shizuoka University (2016).
[5] Iguchi Yu and Takahiro Koita, Consideration on Parameter Setting Scheme for Amazon EC2, IEICE Technical Report, ICM2017-66, pp.61-67 (2018).

Takahiro Koita received PhD degree in engineering from Nara Institute of Science and Technology in Japan. He is currently an associate professor at Doshisha University in Japan. He was a visiting professor at University of Wisconsin-Madison in computer science department. His research interests include cloud computing, human computation, distributed processing and high-performance computing.

Yu Iguchi earned her BS degree in engineering from Doshisha University in Japan. Her research interests include information systems, information technology, and cloud computing. She has published papers in cloud computing and its application. She is a member of the Institute of Electronics, Information and Communication Engineers in Japan.