2023 2nd International Conference on Computing, Communication, Perception and Quantum Technology (CCPQT) | 979-8-3503-4269-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/CCPQT60491.2023.00021

A Test Method for Shortening Test Time of Cloud Computing Platforms

1st Yuanrui Zhu*
Cloud&Network Operation Technology Department
Research Institute of China Telecom
Guangzhou, China
zhuyr1@chinatelecom.cn

2nd Qingyun Meng
Cloud&Network Operation Technology Department
Research Institute of China Telecom
Guangzhou, China
mengqy@chinatelecom.cn

3rd Deheng Li
Cloud&Network Operation Technology Department
Research Institute of China Telecom
Guangzhou, China
lidh11@chinatelecom.cn

4th Chunyu Shi
Cloud&Network Operation Technology Department
Research Institute of China Telecom
Guangzhou, China
shicy@chinatelecom.cn

5th Jiawei Ren
Cloud&Network Operation Technology Department
Research Institute of China Telecom
Guangzhou, China
renjw3@chinatelecom.cn

6th Yuting Wu
Cloud&Network Operation Technology Department
Research Institute of China Telecom
Beijing, China
wuyt6@chinatelecom.cn

***Abstract*—The reliability of cloud computing platforms is crucial. Due to its numerous levels and components, testing requires a significant amount of time. The existing research aimed at shortening test time has room for improvement. They ignored a situation that not all test steps are necessary, which is our direction for improvement. We design virtual machine feature transformation method to enable it reused across test cases, thus avioding the time spent in frequent creation and deletion virtual machine. In this scenario, we abstract the problem into an Asymmetric Traveling Salesman Problem and use greedy algorithms to minimize the total testing time. According to simulation results, our method has achieved good results.**

Keywords- cloud computing; test; ATSP

I. INTRODUCTION

A. Introduction

In recent years, cloud computing has developed rapidly, but accidents are also increasing. The testing of cloud computing platforms is crucial for reliability [1][2][3]. Due to its numerous levels and components, testing requires a significant amount of time [4][5]. According to the test standard on the open-source organization OPNFV [6], we spent nearly ten hours in the laboratory to complete a test of the Infrastructure layer. Depending on the scale of the cloud computing platform, this test time will be increased several times.

Existing research for reducing test time has ignored the issue that not all test steps in a test case are necessary. Test cases aimed at testing virtual functionality typically start with creating a virtual machine and end with deleting it. If virtual machines can be reused across test cases, these frequent creation and deletion steps are no longer needed.

However, test cases have different requirements for virtual machines. Before reusing virtual machines, their features needs to be transformed. According to the requirements of the test cases, we established a feature model of the virtual machine and established a matrix for their transformation. Based on these works, the problem of shortening testing time has been successfully abstracted as an ATSP (Asymmetric Traveling Salesman Problem), which other research has not achieved.

In our method, many excellent solutions to ATSP problems can be applied to shorten testing time. We selected the greedy algorithm from among them and implemented an experiment. The experiment has achieved good results.

B. Related Work

The research aimed at shortening testing time can be divided into two categories: automated testing and selective execution of test cases.

The automated testing, from the perspective of reducing test time, is actually improving the operating speed of testers. There are many related studies. Small-scale test tools include Jmeter for automated load testing [7] and DestroyStack for automated fault injection testing [8]. Large-scale test framework includes framework for functional testing designed by Askhat Nuriddinov [9], configurable framework designed by Kejiang Ye [10]. We have also developed an automated test framework that integrates many test tools [11]. Automated testing can effectively shorten testing time, and our previous experiments have also proven this. However, this approach only improves operation speed without considering whether the operations in the test cases are necessary. In fact, there is room for improvement.

The selective execution of test cases is skiping unnecessary test cases, thus shortening the test time. Chia Hung Kao has designed a method which can judge the condition of SUT(System under test) based on the results of the test case [12]. According to the condition of SUT, it determines the set and sequence of test cases to be executed subsequently. Hanyu Pei introduced DRT theory to design a method [13]. Also based on the results of test cases, it judges which subsequent test cases have a higher probability of finding SUT problems. These mothods are most suitable for finding more problems in the shortest time. For cloud computing platform, more problems are not enough. We need to find all the problems as far as

979-8-3503-4269-7/23/$31.00 ©2023 IEEE
DOI 10.1109/CCPQT60491.2023.00021

possible to ensure the cloud computing platform reliability [14]. In this scenario, skipping test cases should be very careful, which will lead to the weakening of the effect of these methods.

II. METHODOLOGY

A. The Improvement Ideas

The existing methods ignore a situation that not all test steps are necessary. We analyzed more than 80 test cases. The test cases related to virtual functions are basically completed using virtual machines. These test cases are designed to start with the step of creating virtual machines and end with the step of deleting them, as shown in Figure 1. In fact, these steps are preparing for testing. If the virtual machine can be reused across test cases, these steps are no longer necessary. The problem is that because test cases have different requirements for virtual machines, the above idea cannot be directly implemented.

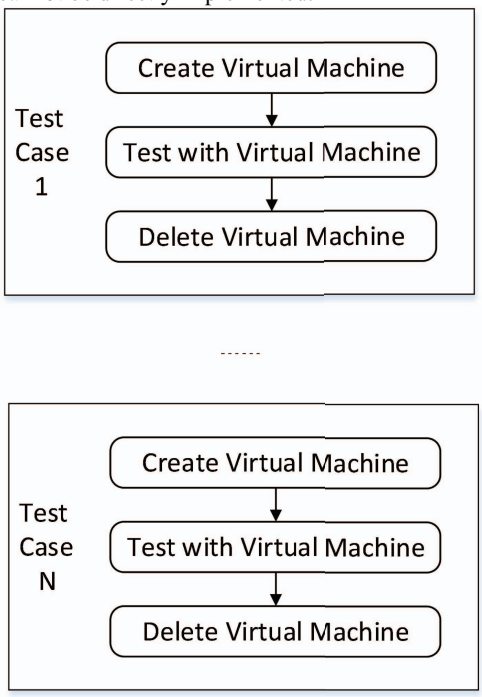

Figure 1. Testing flow chart without reuse virtual machine

The idea of our research is to define the features of virtual machines based on test case requirements. Test cases can be viewed as inputting a virtual machine with A1 feature and outputting a virtual machine with B1 feature. We add step between test cases to convert this virtual machine with feature B1 into feature A2 required for the next test case. This method can implement virtual machine reuse. In this scenario, the first test case starts with creating a virtual machine, and after the test case is completed, the virtual machine is not deleted. Through feature transformation, the virtual machine is passed on to the next test case for use. By analogy, the virtual machine is not deleted until the last test case uses it, as shown in Figure 2.

Implementing virtual machine reuse does not mean reducing testing time, as we have introduced new steps. We abstract the problem of shortening testing time in virtual machine reuse scenarios as a travel salesman problem. Based on this, we established a test case orchestration method and used greedy algorithms to find the optimal solution. These will be described next.

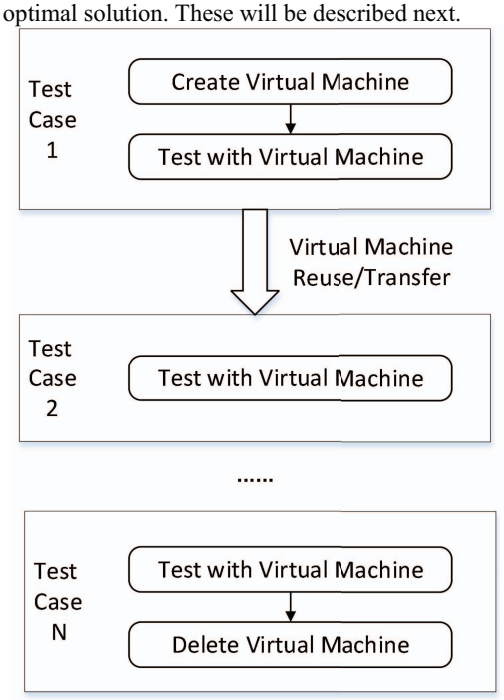

Figure 2. Testing flow chart with reuse virtual machine

B. Virtual Machine Feature Model and Transformation Matrix

1) Virtual Machine Feature Model. Based on the differences in test case requirements, we classify virtual machine features. They are divided into compute, storage, and network features, including the running status of the virtual machine, network card type, CPU number, memory capacity, disk capacity, and so on, as shown in Table1.

TABLE I. VIRTUAL MACHINE FEATURE TABLE

Type	Feature	Feature Values	Transfer Method
Compute	vm state	active, stopped, paused, …	change vm state by api
	host	depends on SUT	migrate vm
	…	…	…
Network	network type	srivo network, dpdk network, …	change vm network card by api
	…	…	…
Storage	storage type	FCSAN, IPSAN, …	change vm disk by api
	…	…	…

2) Virtual Machine Feature Transformation Matrix

a) Single Feature Transformation. Single feature transformation means the value transformation of a single feature, such as a virtual machine state feature changing from active to stop.

A single feature transformation model can be described as an N * N matrix. We have defined two matrices, the transition time matrix T is used to record the time spent in the transformation, as shown in (1). The transition method matrix M is used to record the transformation method, such as the API or command, as shown in (2).

$$\mathrm{T} = \begin{pmatrix} 0 & T_{1,2} & \cdots & \cdots & T_{1,N} \\ T_{2,1} & 0 & \cdots & \cdots & T_{2,N} \\ \vdots & \vdots & 0 & \vdots & \vdots \\ T_{N-1,1} & \cdots & \cdots & 0 & T_{N-1,N} \\ T_{N,1} & \cdots & \cdots & T_{N,N-1} & 0 \end{pmatrix} \quad (1)$$

$$M = \begin{pmatrix} \text{null} & M_{1,2} & \cdots & \cdots & M_{1,N} \\ M_{2,1} & \text{null} & \cdots & \cdots & M_{2,N} \\ \vdots & \vdots & \text{null} & \vdots & \vdots \\ M_{N-1,1} & \cdots & \cdots & \text{null} & M_{N-1,N} \\ M_{N,1} & \cdots & \cdots & M_{N,N-1} & \text{null} \end{pmatrix} \quad (2)$$

$T_{i,j}$ is the time taken for the transition from the i state to the j state. $M_{i,j}$ is the transition method for the transition from the i state to the j state. There is no need to convert between the same feature values, so $T_{i,i}$=0 and $M_{i,i}$ is also a null step. If the i state cannot be converted to j, then $T_{i,j} = \infty$ and $M_{i,j}$ does not exist.

Because it is possible that feature i from k to j are more efficient than feature i to j, the T-matrix will be optimized using the Dijkstra algorithm, and M-matrix will also be updated synchronously.

b) Multi Feature Transformation. The feature of a virtual machine is a collection of P single features $\{ F_1, ..., F_k, ..., F_P \}$, which depends on the number of features involved in the test cases.

Virtual machine feature transformation is the superposition of single feature transformations. For example, a virtual machine with a CPU count of 1 on host A needs to be converted to a virtual machine with a CPU count of 2 on host B. The transformation time is the time when the host feature is from A to B, plus the time when the CPU number feature are is 1 to 2. The transformation method is also similar.

The transformation time of a virtual machine from a feature VM to a VM' is calculated as shown in (3).

$$\mathrm{TransferTime}(\mathrm{VM}, VM') = \sum_{k=1}^{P} T_{F_k,F_k'} \quad (3)$$

In (3), F_k and F_k' is the Kth feature value of the virtual machine feature VM and VM', respectively. $T_{F_k,F_k'}$ is the time taken to move from F_k to F_k', which can be queried from the matrix T.

In order to reduce the testing time, when the transformation time is greater than the time spent in the step of deleting and creating virtual machine , the transformation method and time should be replaced with this step and the time it takes.

C. The Method of Shortening Test Time Based on Virtual Machine Reuse

In our method, the unnecessary steps of creating and deleting virtual machines are replaced. In order to distinguish, we call the remaining steps necessary test steps.

One test including N test cases can be divided into four parts.

- One time of create virtual machine step, at the beginning of the test.
- N test cases' necessary test steps.
- N-1 times of virtual machine feature transformation steps.
- One time of delete virtual machine step, at the end of the test

The calculation method of total test time is as shown in (4).

$$\begin{aligned} &\mathrm{TotalTestTime} \\ &= \mathrm{CreateVMTime} \\ &+ \sum_{i=1}^{N} NecessaryTestStepTime_i \\ &+ \sum_{i=1}^{N-1} TransferTime\left(VM_{END_i}, VM_{START_{i+1}}\right) \\ &+ DeleteVMTime \end{aligned} \quad (4)$$

VM_{END_i} is the output virtual machine feature of the i test case, and $VM_{START_{i+1}}$ is the input virtual machine feature of the i+1 test case. $TransferTime\left(VM_{END_i}, VM_{START_{i+1}}\right)$ is the feature transformation time from the former to the latter.

To further simplify the calculation, we define an empty test case. The empty test case has no steps, and the test time is zero. It uses a virtual machine that does not exist. The characteristics of this virtual machine are defined as null.

Adding two empty test cases before and after the test will not change the total test time. After adding, CreateVMTime can be regarded as the virtual machine feature transformation time from the empty test case to the first test case. DeleteVMTime can be regarded as the virtual machine feature transformation time from the last test case to the empty test case. The simplified calculation method is as shown in (5).

$$\begin{aligned} &\mathrm{TotalTestTime} \\ &= \sum_{i=0}^{N+1} NecessaryTestStepTime_i \\ &+ \sum_{i=0}^{N} VMTransferTime\left(VM_{END_i}, VM_{START_{i+1}}\right) \end{aligned} \quad (5)$$

$\sum_{i=0}^{N+1} NecessaryTestStepTime_i$ does not change, and $\sum_{i=0}^{N} VMTransferTime\left(VM_{END_i}, VM_{START_{i+1}}\right)$ changes with the order of test case orchestration. Therefore, to shorten the total test time is to find the optimal test case orchestration order, which is actually an Asymmetric Traveling Salesman Problem.

We build a directed complete graph, as shown in Figure 3. Point i is the $NecessaryTestStepTime_i$. Point 0 refers to the empty test case, which is the beginning and end of the test. The line segment from point i to point j refers ot the transformation of the output virtual machine feature of test case i into the input virtual machine feature of test case j. The length of it is $VMTransferTime\left(VM_{END_i}, VM_{START_j}\right)$ .The length of each line segment in the graph can be queried based on the T matrix.

The above minimum test time problem is to find a shortest path. This path can start at any point in Figure 3 and end after passing through all other points. The total path distance is $\sum_{i=0}^{N} VMTransferTime\left(VM_{END_i}, VM_{START_{i+1}}\right)$, and this path is the optimal test case orchestration order we are looking for.

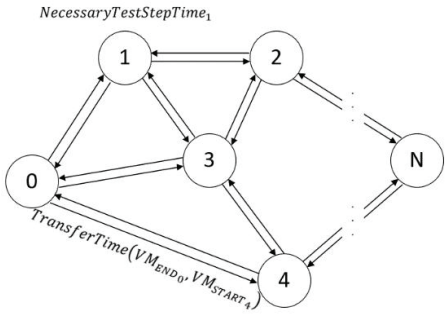

Figure 3. Directed Complete graph Diagram of Test Case Sequence

The method of solving the ATSP problem is not the focus of this study. There have been many studies to solve this problem, and we choose the greedy algorithm from among them. The solution process will not be described in detail. After finding the optimal path, execute the test case according to it. The virtual machine feature transformation method between test cases can be queried based on the M matrix.

III. Experiment Design

A. Selection of test cases

To verify the effectiveness of this method, we designed a simulation experiment. We selected regular test cases as the test case library. We select 5 test cases for simulation each time and compare the total testing time before and after using this method. The selection of test cases is shown in Table2.

TABLE II. Test Case Library Table

	Test Case	Input virtual machine feature	Output virtual machine feature
1	Virtual Machine General Operations	any	same as input
2	Resize Virtual Machine flavor	any	CPU num: 1 Mem size: 4GB Disk size: 10GB
3	Virtual Machine Volume QOS	Volume Qos: 100KB/s	same as input
4	DPDK virtual network supports giant frames	Network type: DPDK Network Mtu: 9000	same as input
5	Virtual Machine Network QOS	Network Qos: 1MB/s	same as input
6	Resize Virtual Machine CPU and memory online	CPU num: 1 Mem size: 4GB	CPU num: 2 Mem size: 8GB
7	Virtual machine CPU binding	CPU type: binding	same as input
8	Virtual machine SRIOV network bridging	Network type: SRIOV	same as input
9	Virtual Machine Huge Page Memory	Mem type: huge page	same as input
	…	…	…

Based on historical data, we calculated the testing time for each test case and the time it took for the output virtual machine to transition to the required feature for the next test case, as shown in Table3.

TABLE III. Test Case Time Table

Test case 1	
test time with necesary steps	546s
tranfer time to test case 2	0s
tranfer time to test case 3	0s
…	
Test case 2	
test time with necesary steps	598s
tranfer time to test case 1	0s
tranfer time to test case 3	127s
…	
Test case 3	
test time with necesary steps	710s
tranfer time to test case 1	0s
tranfer time to test case 2	216s
…	
Test case 4	
test time with necesary steps	7150s
tranfer time to test case 1	0s
tranfer time to test case 2	105s
…	

B. Experimental method

The experiment is divided into three steps:

- Randomly select 5 test cases from the test case library
- Simulate the testing method without using this method, calculate the testing time of the each test case as the total testing time
- Simulate the testing method with using this method, use greedy algorithm to determine the execution order of test cases, and calculate the total testing time according to formula 1

- Repeat steps 1-3 ten times and record the testing time for each time

IV. RESULTS

We select 5 test cases each time and conduct 100 simulation tests. The comparison of testing time before and after using this method is shown in the Figure 4. According to simulation results, this method can shorten the testing time by 20-50%.

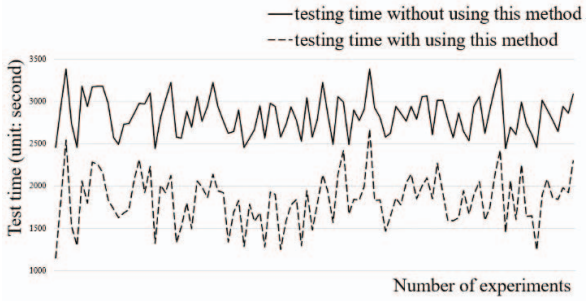

Figure 4. Test time comparison table

V. CONCLUSION

This research aims to shorten the total testing time by reusing virtual machines across test cases. For this purpose, we designed a virtual machine feature model and transformation matrix, and abstracted this scenario into ATSP. We used greedy algorithms to find the optimal solution and designed simulation experiments. According to the simulation results, our method has achieved good results.

REFERENCES

[1] S. Meng, L. Luo, X. Qiu and P. Sun, "A Reliability Optimization Framework for Public Cloud Services based on Markov Process and Hierarchical Correlation Modelling," 2021 7th International Symposium on System and Software Reliability (ISSSR), Chongqing, China, 2021, pp. 86-90, doi: 10.1109/ISSSR53171.2021.00034.

[2] N. Luo and Y. Xiong, "Platform Software Reliability for Cloud Service Continuity - Challenges and Opportunities," 2021 IEEE 21st International Conference on Software Quality, Reliability and Security (QRS), Hainan, China, 2021, pp. 388-393, doi: 10.1109/QRS54544.2021.00050.

[3] R. Zhang, P. Sun, S. Tian and W. Liu, "A Survey on the Application of Cloud Models in the Reliability of Cloud Environments," 2022 8th International Symposium on System Security, Safety, and Reliability (ISSSR), Chongqing, China, 2022, pp. 179-180, doi: 10.1109/ISSSR56778.2022.00037.

[4] R. Sharma and R. Singh, "Reliability based micro-economic cost model for cloud computing systems," 2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), Greater Noida, India, 2021, pp. 293-297, doi: 10.1109/ICCCIS51004.2021.9397210.

[5] S. Meng, L. Luo, X. Qiu and Y. Dai, "Service-Oriented Reliability Modeling and Autonomous Optimization of Reliability for Public Cloud Computing Systems," in IEEE Transactions on Reliability, vol. 71, no. 2, pp. 527-538, June 2022, doi: 10.1109/TR.2022.3154651.

[6] T. Kim, T. Koo and E. Paik, "SDN and NFV benchmarking for performance and reliability," 2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS), Busan, Korea (South), 2015, pp. 600-603, doi: 10.1109/APNOMS.2015.7275403.

[7] R. K. Lenka, S. Mamgain, S. Kumar and R. K. Barik, "Performance Analysis of Automated Testing Tools: JMeter and TestComplete," 2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), Greater Noida, India, 2018, pp. 399-407, doi: 10.1109/ICACCCN.2018.8748521.

[8] Arlat J, Aguera M, Amat L, et al. Fault injection for dependability validation: A methodology and some applications[J]. IEEE Transactions on Software Engineering, 1990, 16(2): 166-182.

[9] A. Nuriddinov, W. Tavernier, D. Colle and M. Pickavet, "A framework for functional testing of VNFs," 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Verona, Italy, 2018, pp. 1-2, doi: 10.1109/NFV-SDN.2018.8725698.

[10] K. Ye, J. Che, X. Jiang, J. Chen and X. Li, "vTestkit: A Performance Benchmarking Framework for Virtualization Environments," 2010 Fifth Annual ChinaGrid Conference, Guangzhou, China, 2010, pp. 130-136, doi: 10.1109/ChinaGrid.2010.12.

[11] Zhu, Yuanrui, et al. "An efficient automatic testing framework for NFV systems." International Conference on Cloud Computing, Performance Computing, and Deep Learning (CCPCDL 2022). Vol. 12287. SPIE, 2022.

[12] C. H. Kao, P. -H. Chi and Y. -H. Lee, "Automatic Testing Framework for Virtualization Environment," 2014 IEEE International Symposium on Software Reliability Engineering Workshops, Naples, Italy, 2014, pp. 134-135, doi: 10.1109/ISSREW.2014.28.

[13] H. Pei, B. Yin, M. Xie and K. -Y. Cai, "A cloud-based dynamic random software testing strategy," 2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), Singapore, 2017, pp. 509-513, doi: 10.1109/IEEM.2017.8289943.

[14] B. -A. Tong, X. Li and L. Xiao, "Service Reliability Oriented Modeling for the Failure of Cloud Data Center," 2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), Lisbon, Portugal, 2018, pp. 353-357, doi: 10.1109/QRS-C.2018.00068.