A Reliable and Efficient Stress Generation and Control System

Yan Gong and Lin Huang

***Abstract*—With a large number of concurrent users, many large-scale software systems will face a heavy computational load. In this case, a stress test to verify the performance of large-scale software systems becomes especially important. This paper builds up software system stress generation model, designs and implements a new stress generation and control system, which simulates high concurrent stress requests. It provides control of the stress generation model, scheduling of test scripts, and statistical functions of test information. The results show that the stress generation and control system support the testing of the database and large-scale software system. The simulation has a large amount of concurrency, fast execution speed and low resource consumption.**

***Index Terms*—Stress test model, control system, large-scale software systems.**

I. Introduction

Stress testing is the guarantee of the basic quality of large-scale software systems [1]. The basic idea of stress testing is to let the system under test run for a long time or a large load by initiating a large number of simulated user request operations [2]. And then analyze the performance of the system by analyzing the use of the system resources and the response time of the message The stress generation and control system continuously applies stress to the system until it reaches the unacceptable performance point of the system, thereby judging the system bottleneck and determining the maximum service level that the system can provide. In other words, the stress test uses a concurrent method to increase the system load until it fails and obtain a series of indicators of the system. thereby obtaining the maximum processing capacity of the system. TPS (Transaction Per Second) refers to the number of things or transactions that a system can process per second. It is a very important indicator to measure the processing power of a system [3], [4].

The request stress that the large-scale software system bears in the actual scenario changes with time. Therefore, in order to simulate the user's actual application behavior habits, it is necessary to establish a stress generation model according to the change law of the actual stress of the large-scale software system [5]. Per the generation model, the stress request value initiated to the system is changed, and the stress resistance capability of the system is scientifically verified.

Common testing tools include LoadRunner [6], JMeter [7], WebLOAD [8], etc. they will discover the system's bottleneck by simulating the processing power of the entire system by simulating thousands of user requests. However, when these test tools use the timeout mechanism when sending the request message, that is, when the request time exceeds the present time, the virtual user stops sending the request to the system in the future, so that the stress of the system under test becomes smaller and smaller.

This paper designs and implements a reliable and efficient stress generation and control system that provides control of the stress generation model and can simulate high concurrent stress requests. The remaining chapters of this paper are organized as follows: Section II describes the stress generation model, Section III describes the stress generation and control system we designed and implemented, Section IV is the experimental construction and experimental results, and Section V is the summary paper.

Manuscript received December 30, 2018; revised March 25, 2019.

The authors are with the Academy of Military Sciences PLA China (e-mail: gongyan2007@gmail.com, huanglinwxs@163.com).

II. Stress Generation Model Design

The model of stress generation mainly includes two aspects, one is the change model of the whole stress during the test process; the other is the control of the stress generation mode in the second period. In view of the above-mentioned content, different stress generation models need to be designed, and the model is controlled in the test to find out the system bottleneck more accurately and efficiently and obtain various performance indexes of the system under test.

This paper proposes the following models for the overall stress change during the test process. The following will be a detailed analysis.

A. Constant Stress Generation Model

The simulation request is performed at a fixed stress, and the strength of the stress during the test is always constant. The control of this model is relatively simple, and the stress cannot be dynamically adjusted during the test.

B. Linear Growth Model

Set the initial stress, the growth rate, and the termination stress. Starting from the initial stress, the stress of the simulation request is gradually increased according to the growth rate. After the maximum stress is reached, the stress change curve is as shown in Fig. 1. The control of this model is relatively complicated. As the stress increases, the system under test may have problems such as long response delay and reduced processing success rate.

We use T to represent the stress value TPS corresponding to the model; use T_S to indicate the preset starting stress; k to represent the rate of change of stress; t_n and t_s represent the current time of the test and the start time of the test,

doi: 10.18178/ijfcc.2019.8.2.537

respectively; P represents the period of change of the test. When the test stress does not reach the maximum stress, the current stress is expressed by the formula (1), and when the stress reaches the maximum value, the maximum value is used to indicate the current stress.

$$T = T_S + k(t_n - t_s) / P \quad (1)$$

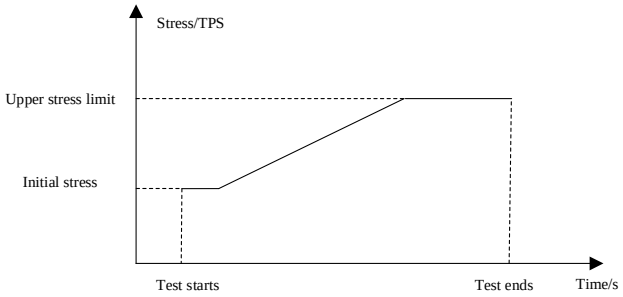

Fig. 1. Linear growth model stress curve.

C. Dynamic Change Model.

Set a change period, set a range of variation for each period, and randomly generate a TPS value within a preset range of change after each period. The stress value changes randomly within a certain range with time, without a fixed pattern, which is relatively consistent with the access situation of long-term real user's. it is used to simulate the change of TPS in the design scene at different times of the day. The stress curve is shown in Fig. 2.

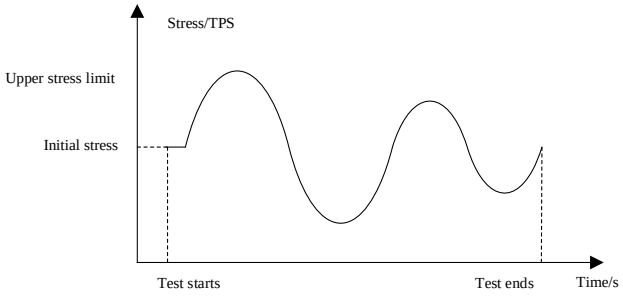

Fig. 2. Dynamic change model stress curve.

We use T to represent the stress value TPS corresponding to the model; use T_S to represent the preset starting stress; use the function to represent the random number in the interval. Then the value of the dynamic change model stress TPS is determined by the formula (2). Indicates that the stress is randomly floating up and down around the initial stress.

$$T = T_S + Random(R_{\min}, R_{\max}) \quad (2)$$

Following methods are proposed for the stress generation mode in smaller granularity in seconds.

1) Explosive. The amount of concurrency at the beginning of each second is generated at the same time, and the request for the same test concurrency is regenerated in the next second.
2) Gentle. The test concurrency per second is generated smoothly over a period of 1 second, with an average fraction per second or smaller, a certain amount of stress per small unit time, and the same test is generated in the same way in the next second with same amount of concurrency stress.
3) Queued. The test maintains a certain level of concurrency. When the concurrency limit is reached, no new test is initiated. A new test will be ran after the on-going one is finished. That is to say, when the test generates a new stress request, it first checks the number of stress requests that have been initiated, and if it does not reach the set TPS value, new request won't be initiated; if the set TPS value is not reached, a new request will be initiated. The requested amount is filled to the set TPS value.

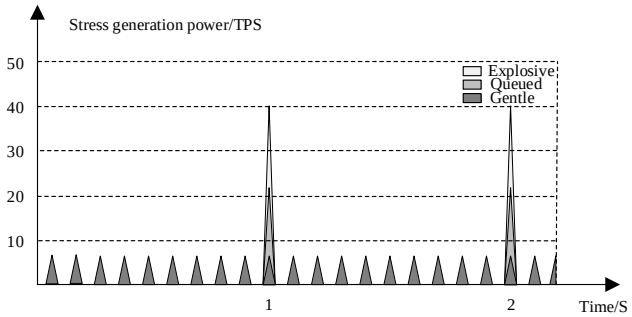

Fig. 3. Comparison of three stress generation methods.

Fig. 3 shows the changes in stress generated by the three stress generation modes. The lightest color indicates the explosive stress generation mode. It can be seen that the instantaneous stress is generated at the beginning of each second; the dark gray indicates the gentle stress generation mode, and the stress is generated at ten-time points in one second, which is generated. The stress curve is relatively flat; the gray is the queue-type stress generation method. It can be seen that the stress is not the same at several time points per second. The reason is that there are several requests in one second which have already been responded. So the stress generated by the gray is only to fill the lack.

III. Stress Generation and Control System

The stress generation model has been designed. Following section will discuss how to use it in a stress control system. The stress generation model is saved to the stress control system through the configuration file before the test starts, and the key information of the model is written into the shared memory when the process starts. At the same time, a field is maintained in the shared memory to indicate the stress TPS value that needs to be generated at present. Then, according to the stress generation model, the stress value is periodically calculated and updated into the shared memory. When the test is triggered, the stress will be triggered per the value in the memory, thus achieving the purpose of controlling the stress generation model. The stress is generated in seconds, and a cycle program is added to the main program of the execution system to generate the required stress at a fixed time point, according to the stress generation mode on the second level.

The stress generation and control system is a core functional component of the stress test platform. An

automaton that simulates a stress request is generated using a scheduler while communicating with the system via communication module. According to its different functions, the stress generation and control system consists of four functional modules: task scheduling, stress control, concurrency control and statistical control. The relationship among modules is shown in Fig. 4.

A. Task Scheduling Module

The task scheduling module is the core part of the whole stress generation and control system to realize the allocation and scheduling of each function of the test. The main work flow is:

1) Accepting the control command;
2) Calling the stress control module and controlling the stress generation model;
3) Call the concurrency control module to apply the second-level service trigger mode;
4) Select the test script and schedule the automation to achieve the purpose of generating stress on the system under test.

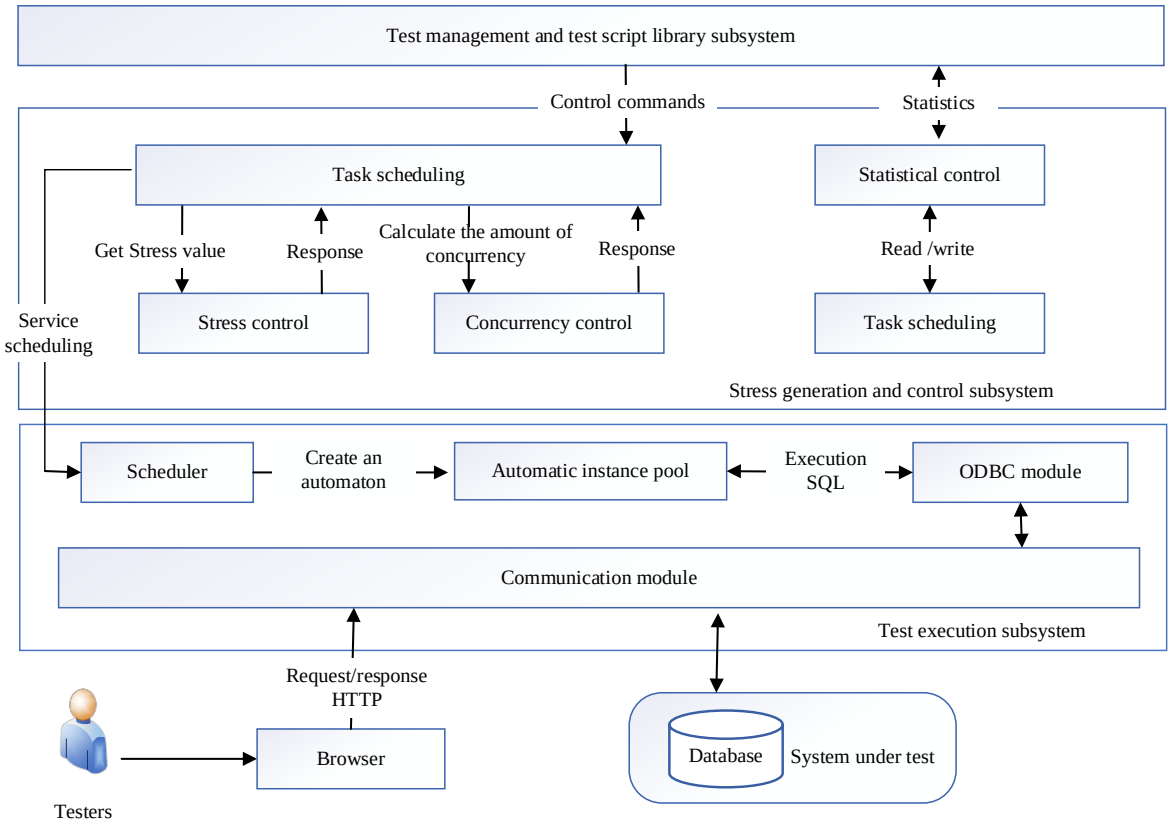

Fig. 4. Stress generation and control system module diagram.

B. Stress Control Module

1) Constant stress generation model. When the stress control module checks the stress control information in the shared memory, if the set stress generation model is constant, there is no need to change the stress, and the stress will always maintain the initial set stress.
2) Linear growth model. The linear growth model has the following key parameters: initial stress value, maximum stress value, stress change period, and step size of stress change. At the beginning of the test, the stress is set to the initial value, and then the stress is increased in every cycle until the end of the test or when the ultimate stress is reached. When the stress control model is a linear growth one, the stress is increased by cycle. When the stress exceeds the maximum value, it is set as the maximum value.
3) Dynamic change model. The main parameters of the dynamic change model are: the initial value of the stress, the period of change, the range of variation, and the stress changes around the initial value during the test. At the beginning of the test, the stress value is set to the initial value, and then each cycle randomly generates a change within the variation range. The initial value is used as standard, and the amount of change is increased or decreased to achieve the stress value of the cycle.

C. Concurrency Control Module

The concurrency control module controls the second-level stress generation mode and generates corresponding concurrency control information according to the concurrency mode established by the test. The concurrency control module obtains the stress control information from the shared memory and calculates the concurrent stress TPS according to the time of the test, and returns to the task scheduling module. The concurrency control module is periodically scheduled by the task scheduling module and returns the stress value required to be generated in the current cycle per the stress generation mode of the second level, the task scheduling module performs the scheduling of the test script according to the stress value. The second-level stress generation mode is calculated as follows.

1) Explosive. The concurrency control module only needs to check in seconds, and whenever a new second starts, new stress is generated. That is to say, the concurrency control module checks the current time. If the test enters a new second, the stress value TPS is read from the shared memory and returned directly, indicating that stress has to be generated at the instance of new second.
2) Gentle. The concurrency control module subdivides one second into ten milliseconds. When the test time enters a new 100 milliseconds, it returns one tenth of the stress that needs to be generated in this second (TPS).
3) Queued. the TPS value of the stress control module minus the currently generated one, then the TPS that need to be generated in this scheduling will be obtained.

D. Statistical Control Module

The statistical control module is mainly responsible for the information in the statistical testing process. The test business script records the start time, end time, and run result of the test during the test. The statistical control module first counts the total number of requests sent, the total duration of request processing, the maximum delay, and the minimum delay, and calculates the average response delay based on the total amount of transmission and the total execution time. The average response delay is further divided into average delay for processing success, the average delay for processing failures, and the usage of CPU and memory at the corresponding time points. Then, according to the user configuration, the statistics of the delay in a certain period are also collected, including the total number of transmissions, the number of successes, the number of failures, the maximum delay, the minimum delay, and the average delay.

IV. Experimental Validation

A. Test Environment

The test environment is shown in Fig. 5. One PC is used as the user operation host (Intel CoreTM2 P8400 2.26GHz, 3GBDDR), two servers are used to deploy the stress generation node (Intel 2x4 Core X5450 3.00GHz, 4GBDDR), and two servers are used to deploy the tested application system and Database (Intel 2x4 Core X5450 3.00GHz; 4GBDDR, RedHat Enterprise Linux 6.0, DM7.0). The total number of database table records exceeds 100,000, and the number of fields per record is 45. The field types include integer, character, and float. The total database record capacity is 8G.

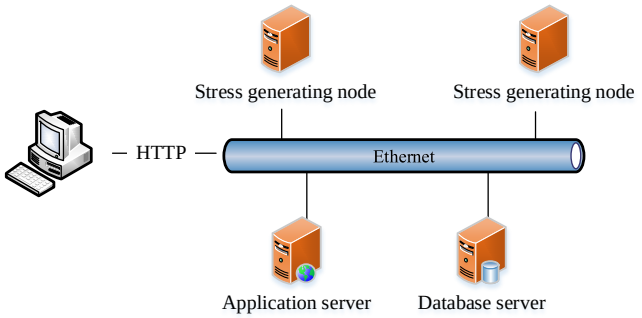

Fig. 5. Test environment.

B. Performance Testing

The performance of the stress generation and control system mainly refers to the ability to simulate the concurrent stress, so it can be tested by simulating the deployment of a real test environment and gradually increasing the number of simulation requests.

The performance test focuses on the following indicators:

1) Message processing delay. Message processing delay is an important indicator to measure the concurrent capability. When the amount of the request is large, the transmission needs to be completed in a short time to avoid the stress generation and control system because of the transmission delay. Sometimes it affects the accuracy of the test.
2) Stress generation and control system load situation. Large transmission stress will also cause a huge burden on the stress generation and control system. Each node has multiple processes, and the load conditions of each process must be guaranteed to be generally the same. The stress is accumulated on one or several processes, affecting the transmission efficiency.
3) Resource occupancy rate. If the stress generation and control system consume too much CPU resources and memory resources when processing a large amount of messages, it may cause an increase in message transmission delay, and may even result in loss of messages, which makes stress generation and control system in an unstable state.

The test results as shown in Table I, from the performance index data, it can be seen that the stress generation and control system has a strong ability to process the transmission of batch messages. The performance can also be extended horizontally by the expansion of the number of stress generating nodes.

TABLE I: Performance Test Results

Node	Concurrency	Delay	CPU usage	Memory usage
Single node	20000tps	0.48ms	Average 52% Peak 63%	Average 58% Peak 66%
Double node	30000tps	1.3ms	Average 56% Peak 63%	Average 58% Peak 68%

C. Stability Test

In general, an indicator of the stability of a system is a stable operation that can run without problems for 7 x 24 hours. Therefore, the stress generation and control system also needs to ensure stable operation for at least 7 x 24 hours. The specific test method is as follows. For different single test operations, the set stress is cyclically changed, and the stress value is about 80% of the limit capability. The stress request is continuously initiated to a system under test, which is caused by the stability problem of the system under test. The test is terminated, so the system under test can only receive messages and respond, without actual operation. The test results show that the stress generation and control system can continuously apply stress for more than 15 days, the log records are normal, the test report is normal, and the stability requirements are met.

D. Cross-Platform Capability Test

Cross-platform testing means cross-platform testing of stress generation and control systems and cross-platform testing of web pages. The resource generation and performance of the stress generation and control system on different platforms, and whether it can run stably, the test results are shown in Table II. It can be seen from the test results that on the Kylin4 operating system, although the resources are relatively large, it can also meet the basic functional requirements.

TABLE II: CROSS-PLATFORM ABILITY TEST RESULTS

Platform	Concurrency	CPU usage	Memory usage
RedHat 6.0	20000tps	52%	58%
Kylin4	15000tps	72%	86%

Web page cross-platform testing is mainly reflected in the comparison between different platforms and different browsers. Usually IE browser only supports Window operating system, so no need to do the comparison. As for the comparison of pages, it is mainly compared on browsers such as FireFox, Chrome and Opera. The test results are shown in Table III.

TABLE III: WEB PAGE CROSS-PLATFORM TEST RESULTS

Browser	Page open speed
FireFox	0.46s
Chrome	0.21s
Opera	0.30s

It can be seen from the test results that the web page of the stress generation and control system is applied on each browser, and the speed is slightly different, and the display and function are normal.

V. CONCLUSION

The generation of stress is not an irregular transmission of a series of simulation requests, and it is necessary to follow certain rules in order to measure the system under test in a more scientific way. This paper designs and implements a reliable and efficient stress generation and control system that provides control of the stress generation model and can simulate high concurrent stress requests. The experimental results show that the stress generation and control system supports the testing of the database and the application system. The concurrent simulation capability is strong, the message processing time is low, the resources are minimal, and the system runs stably and reliably.

REFERENCES

[1] M. J. Zhen and E. H. Ahmed, "A Survey on load testing of large-scale software systems," *IEEE Trans. Softw. Eng*, vol. 41, no. 11, pp. 1091-1118, 2015.

[2] J. A. Meira, E. C. Almeida, and G. Sunye, "Stress testing of transactional database systems," *Journal of information and Data Management*, vol. 4, no. 3, pp. 279-294, 2013.

[3] S. Nejati, S. D. Alesio, and M. Sabetzadeh, "Modeling and analysis of CPU Usage in safety-critical embedded systems to support stress testing," in *International Proc. of Model Driven Eng. Languages Syst*, 2012, pp. 759-775.

[4] O. Krejcar and L. Motalova, "Home care web services evaluation by stress testing," in *International Proc. of the e-Technologies and Networks for Development*, 2011, pp. 238-248.

[5] D. Krishnamurthy, J. Rolia, and S. Majumdar, "A synthetic workload generation technique for stress testing session-based systems," *IEEE Trans. Softw. Eng*. vol. 32, no. 11, pp. 868-882, 2006.

[6] HP LoadRunner software. (2018). [Online]. Available: https://software.microfocus.com/zh-cn/products/loadrunner-load-testing/overview/

[7] Apache JMeter. (2018). [Online]. Available: http://jakarta.apache.org/jmeter/

[8] WebLOAD product overview. (2018). [Online]. Available: http://www.radview.com/

Yan Gong was born in Beijing, China in 1980. He received his Ph.D. degree in computer science at Beijing University of Posts and Telecommunications of China in 2010. He is now the senior researcher in the Academy of Military Sciences PLA China. His main research interests include service computing, software performance engineering and monitoring of distributed system.

Lin Huang was born in Beijing, China in 1980. She received her Ph.D. degree in computer science at Beijing University of Posts and Telecommunications of China in 2018. She is now the senior researcher in the Academy of Military Sciences PLA China. Her main research interests include software performance engineering and mining software repositories.