Research on Performance Testing Technology Based on Scenario Design

Zhang Hui
Gaobo Software College, Nanning College, Nanning, Guangxi, China
Corresponding author email: 2351452593@qq.com

Abstract: With the rapid development of web system application, performance testing is very important to ensure the fault free operation of web system. The scenario design in performance testing is the basis of performance testing. Only reasonable design of test scenarios can obtain valuable test data. This paper starts from the web performance testing, explains the function and design of performance testing scenarios, then analyzes the common performance testing scenarios, and finally carries out web performance testing practice through testing tools. The experimental results show that the performance test of different scenarios can help to find out and solve the main problems in the web system, and provide an effective basis for system optimization.

1. Introduction
With the development of Web services and the increasing number of users, web servers are required to respond to user requests quickly and timely. Web performance test is an important test method to evaluate web system, which aims to ensure the normal operation of the system. Scenario design is a technical means to simulate a large number of user environments and generate load to the server to verify whether the performance indicators of the system meet the user requirements. This paper analyzes the principles and Strategies of different scene design, and through the test practice of two different scene design, we can more effectively find the performance problems of the system under test.

2. Web performance test
2.1 Principle of performance test
Users send response requests to the system online, which will occupy the system's cup, memory, network, database and other resources. If a large number of concurrent transactions are done at the same time, the system resources may not keep up with the consumption. Due to the performance bottleneck of the system, the request sent by the user can not get timely response, or return the wrong response.

Performance test is to test the bearing capacity of the whole system by simulating a large number of users, to achieve stress test, load test, configuration test, peak test and so on. By comparing all kinds of data, this paper analyzes the causes of system performance bottleneck, and reduces the risk of system operation.
2.2 Performance test index

2.2.1 Main indicators

(1) Response time: refers to the response time of the system to the request, and the response time of different functions is also different. It usually refers to the average response time of all functions of the system or the maximum response time of all functions.

(2) Throughput: throughput refers to the amount of traffic handled by the server per unit time. In a certain period of time, the number of users continues to increase, while the throughput tends to be balanced or even slightly reduced without increasing, indicating that the number of users is the maximum number of users that the current system can bear, and the processing capacity of the system has reached the bottleneck.

(3) TPS: refers to the number of transactions processed by the server per second. The larger the TPS value, the stronger the system processing capacity.

(4) Number of concurrent users: simulate multiple users accessing the server at the same time. The number of concurrent users refers to the number of online users.

2.2.2 Index analysis

The following figure 1 illustrates the relationship between the commonly used indicators.

![Figure 1: Index relation curve](image)

When only one user accesses the system, the CPU is idle. With the increase of the number of concurrent users, CPU utilization increases, TPS increases correspondingly, and the average response time also increases, and the increase of average response time is an exponential curve. When the number of concurrent requests is close to the maximum, there are many requests to be processed every second, resulting in frequent process switching. The real time for processing requests is less, and the number of requests that can be processed per second is also less. However, the waiting time of users' requests will become larger, even more than the maximum psychological waiting time of users.

2.2.3 Performance test index algorithm

In order to ensure the quality of the test, the performance test index algorithm usually adopts the two eight law. The law of two eight means that 80% of the business volume is completed in 20% of the time, so as to quantify the business requirements.
For example, if you design a user login scenario, in the morning peak period from 10:00 to 10:20, 5000 users log in online, then the system needs to process 5000 services, and the processing time is $20 \times 60 = 1200$ seconds, then the throughput $= 80\% \times \text{services} / (20\% \times \text{time}) = 4000 / 240 = 16.7/s$, not $5000 / 1200 = 4.1/s$. In fact, the distribution of login requests is a normal distribution. The peak time is definitely higher than 4.1/s. In the peak period, 80% of the traffic is actually completed, but only 20% of the time is spent.

The result calculated by the law of two eight is not the number of online concurrent users, but the processing capacity (throughput) of the system. If the system performance requirements are higher, we can also choose the one nine law or more stringent algorithm.

3. Function and design of performance test scenario

3.1 The role of performance scenarios
Performance test scenario is a kind of pressure test activity for application in order to achieve a specific test goal. The design and implementation of performance test scenario is the core of the whole performance test project activities. Without a complete scenario design, the purpose of testing cannot be achieved. Without a reasonable scenario design, the performance defects of the system cannot be found effectively. Test scripts and test data are prepared to achieve specific performance scenario design.

3.2 Performance scenario design
(1) Clear performance requirements, determine the business application scenarios.
(2) Business scenario modeling is carried out according to specific application scenarios. Confirm business scope confirmation, business operation process, business matching, thinking time, assembly point and other configurations.
(3) Configuration parameter configuration. The configuration of gradient increasing mode, instantaneous pressurization mode and throughput mode are confirmed respectively.
(4) Confirm the relationship between parameterized data and parameter association between interfaces.
(5) Confirm the monitoring index, start stop standard, operation time, etc.

4. Performance test scenario analysis
This paper analyzes several common test scenarios, and studies the principle, purpose and implementation strategy of different performance test scenarios.

4.1 Single business benchmark
(1) Principle: single service benchmark is to simulate a single virtual user to test a certain performance index of one or more service test objects quantitatively and comparably.
(2) Objective: single business benchmark is to obtain the processing time of single business under the condition of no pressure on the server, so as to provide data support for subsequent optimization;
(3) Execution strategy: a single user pressure tests the service for 5-10 minutes to understand the response time and TPS of the service.

4.2 Single service load test
(1) Principle: by simulating concurrent virtual users, the virtual customers increase in gradient level, and each virtual user level makes a separate scene, and continuously circulates for a certain time to obtain transaction response time, TPS, error rate, and monitor the resource utilization of each server of the system.
(2) Objective: to obtain the maximum processing capacity of a single business of the system, as well as the correlation and change trend between performance indicators, such as the change trend of response time with TPS, the change trend of TPS and response time with the number of concurrent users, and the change trend of CPU utilization with TPS.
(3) Execution strategy: single service load test is generally executed in a gradually pressurized way for 30 minutes to observe the performance inflection point. At the same time, monitor the server resources and database processing capacity.

4.3 Mixed service load test
(1) Principle: mix the services that need to be pressure tested according to a certain proportion, take the minimum number of concurrent users required by customers as the benchmark, and increase the number of concurrent users in a certain gradient to pressure test the mixed transactions until the system has a performance inflection point.

(2) Objective: To investigate the trend of the system's processing capacity with the change of load, and analyze the maximum processing capacity of the system.

(3) Execution strategy: allocate according to proportion, conduct 1-2 hours by gradually pressurizing, monitor server resource consumption, database processing capacity, etc.

4.4 Stability test
(1) Principle: stability test is to verify whether the system is stable by loading a certain pressure on the system and running for a long time.

(2) Objective: the processing capacity of the system under extreme load for a long time, whether there are hidden problems such as longer response time, memory leakage, insufficient disk space, etc. with the growth of test time;

(3) Test strategy: 8 hours by gradually pressurizing (according to the actual situation), monitoring server resource consumption, database processing capacity, etc. The load pressure of the stability test can be 70% of the maximum processing capacity of the system or a certain pressure value in the mixed scenario.

4.5 Surge test
(1) Principle: verify the normal operation of the system by changing the system from high load to low load, then climbing to high load and then reducing. Mainly for the uncertain short-term peak traffic inflow scenarios, such as social hot topics.

(2) Objective: to verify whether the system can provide services normally and stably in the case of sudden increase of concurrency or large fluctuation of requests in a certain period of time.

(3) Test strategy: take the number of concurrent users with the optimal processing capacity of the system (the resource utilization rate is close to 75%) as the maximum number of concurrent users, and the number of concurrent users with the resource utilization rate of about 20% - 30% as the minimum number of concurrent users. The minimum number of concurrent users and the maximum number of concurrent users run alternately. Each gradient runs for 10 minutes for a total of one hour.

5. Performance scenario test practice
This time, we take the transaction on the login page of a shopping website as the performance test case, and use jmeter automated testing tool to test. By setting two scenarios of single business benchmark test and single business load test, we compare the corresponding transaction time, throughput and other data, judge the performance and bottleneck of the corresponding modules of the system, and analyze whether the system performance meets the requirements.

5.1 Single business benchmark
(1) Test scenario: single user, running for 5 minutes.

(2) The test results are shown in Table 1.
Table 1: single business benchmark table

| Concurrent number | Running time | Transaction average response time | Transaction maximum response time | TPS (pen/sec) | Transaction successful rate% | Message return failure rate% |
|-------------------|--------------|------------------------------------|-----------------------------------|--------------|------------------------------|-----------------------------|
| 1                 | 5            | 0.0368                             | 1.175                             | 43           | 99.99                        | 0                           |

(3) Data analysis:
When one user runs for 4 minutes, the average response time of each transaction is 0.0368 seconds, which is within the normal range. Server resource usage is within normal range.

(4) Performance evaluation:
In the process of single service benchmark, the resources used by the server are within the normal range, no downtime and memory overflow exceptions are found, and the response time is within the normal range, which meets the performance requirements.

5.2 Single service load test
(1) Test scenario: according to the method of concurrency gradient increase, with 20 concurrency, 40 concurrency For gradient pressurization, check the system for performance defects.
(2) The test results are shown in Table 2.

Table 2 single service load test table

| Concurrent number | Running time (min) | Transaction average response time | Transaction maximum response time | TPS (pen/sec) | Transaction success rate% | Message return failure rate% |
|-------------------|--------------------|------------------------------------|-----------------------------------|--------------|------------------------------|-----------------------------|
| 20                | 5                  | 0.007                              | 0.53                              | 31.8         | 100                          | 0                           |
| 40                | 5                  | 0.305                              | 0.38                              | 31           | 100                          | 0                           |
| 60                | 5                  | 0.601                              | 0.9                               | 32           | 100                          | 1/635                       |
| 80                | 5                  | 2.9                                | 6.1                               | 34.5         | 92                           | 5.15                        |
| 100               | 5                  | 2.8                                | 5.8                               | 39.3         | 88                           | 10.87                       |

(3) Data analysis:
With the increase of concurrency, the response time and the corresponding time of transaction change. When less than 60 users log in at the same time, all performance indicators are in the normal range, the probability of message return failure is small, TPS is stable at about 31, and the maximum response time of transaction is less than 1 second.
But when the number of concurrent users reaches 80 and increases upward, the failure probability of the system return message increases greatly, and other indicators are normal. When the customer needs 100 concurrent users, the message return failure rate reaches 10.87%.
(4) Performance evaluation:
In the single business load test process, the system supports 60 concurrent operations, and the maximum transaction response time is less than 1 second. Within a reasonable range, the transaction success rate can meet the demand. But with the increase of the number of concurrent users, especially the number of concurrent users more than 100, the success rate of transaction is greatly reduced. The load capacity of the system needs to be improved and must be further optimized.

6. Conclusion
With the wide application of web system software, the system needs to bear more and more load and pressure. In order to ensure the quality and reliability of software, web performance testing is a very important part of software testing technology. How to improve the performance of the system and solve
the problems in the actual operation of the system is also the fundamental goal of performance testing. The design of performance testing scenarios is the basis of performance testing. Only by designing different performance testing scenarios according to the actual performance testing requirements, can the efficiency of performance testing be improved, the test cases be closer to the actual use of users, and it is easier to find system bottlenecks, so as to provide solutions for software tuning and provide better software services.

Acknowledgments
Funded project: 2019 Nanning College level scientific research project "Research on performance testing technology based on Web application" (No. 2019xj24)

reference
[1] Yang Yan, Liu Zhao, Cai jiutao. (2020) research and practice of software testing based on exploratory testing. Computer applications and software, 37 (06): 29-33 + 64
[2] Liu Kai, Liang Xin, Zhang Junping. (2018) research on software testing process model. Computer science, 45 (S2): 518-521
[3] Chen Zuo, Zhang Huaixiang, Fang Jinglong. (2018) embedded software automatic testing technology. Computer engineering and design, 2018,39 (10): 3125-3131
[4] Yan Qiangqiang, Ma Jun, Xia Yingzhou. (2020) research on performance test of command information system software based on JMeter [J]. Informatization research, 46 (02): 74-78
[5] Mengqing TanLi,Yan Jiang,Xiang Wang.(2019)Black-box Approach for Software Testing Based on Fat-property[C]//Proceedings of the 2019 International Conference on Computer Science,Communication and Network Security,Sanya:DEStech Publications,Inc.:198-204.
[6] Mengqing TanLi,Yan Jiang,Yulin Wang.(2018)Digital inspection of cutting and machining based on manufacturing quality for shop floor[C]//Proceedings of the 2018International Conference on Mechanical,Electronic and Information Technology ICMEIT2018,Shanghai:DEStech Publications,Inc.:15-16.