Realizing a Distributed Performance Testing System Based on TTCN-3

X.M. Liu
College of Information
Beijing City University
Beijing, China

S.M. Liu
College of Science and Information
Qingdao Agriculture University
Qingdao, China

Y.P. Liu, J. Wu
College of Computer
Beihang University
Beijing, China

***Abstract*—In this paper a distributed testing system is designed, which provides a mechanism of node communication, test script deployment, test scheduling, execution-driving and test result collection in distributed environment. A workload model is established, by which testers can describe the performance testing requirement. A performance testing framework is given, which simulates user behaviours in real environment based on virtual users so as to generate workload from the system under test. It can control the execution of virtual users by TTCN-3 standard interface. After executing the performance testing, test report is generated by extracting log. A method of generating performance test-case is studied by reusing functional test scripts. By executing performance testing on an online bookstore, this paper demonstrates the availability of the method of reusing TTCN-3 functional test scripts and the capability of distributed performance testing system established.**

Keywords-TTCN-3; distributed testing system; performance testing; test suite; system under test

I. INTRODUCTION

TTCN-3(Testing and Test Control Notation) provides a set of unified language standard for different users and tool developers, which can be used as a description language to test the response system of various communication ports [1]. Research on performance testing based on TTCN-3 is very important to broaden the application field of TTCN-3. At present, the performance testing tools based on TTCN-3 are very rarely, so as to simulate the load, it is usually done by adding the network traffic generator on the basic TTCN-3 test system[2]. But it is a network testing tool, which can only test the network delay between two nodes and didn't consider the business logic in a real scenario and other load indexes [3].

In this paper, the distributed performance testing system has been designed, which expands TTCN-3 by using typical performance testing methods. The transparent communication method between nodes is also designed, which can solve many communication problems such as the node addressing, message management, timeout retransmitting, message distribution, event registration, monitoring etc. [4].

II. TTCN-3

TTCN-3 is a testing language based on text. The syntax of its core language is similar with the traditional programming language, but it has special test propagation characteristics. [5-6].The top element TTCN-3 is module which is the basic building block. TTCN-3 is used to describe testing [7]. The test requirement is represented by test-cases and test data. Test-case is the most important behaviour of TTCN-3, which describes the test requirement, depicts the test behaviour logic and specifies the test criteria. In a TTCN-3 testing system, a test-case is allowed to run at the same time [8]. Namely, TTCN-3 doesn't support the concurrency execution between test-cases [9]. The TTCN-3 testing system can be viewed as the entity sets which are interactive [10]. Each entity is associated with a specific function in the testing system. These entities manage many operations, such as executing test, interpreting or executing the compiled TTCN-3 code, realizing the correct SUT, executing the external function, processing the clock and so on [11]. TTCN-3 solves the problem of describing the testing system from two levels, namely, platform-independent and platform-dependent. In the platform-independent level, it focuses on describing the test logic (including test-case, test data, test components), and cannot solve the test execution. In the platform-dependent level, it focuses on describing how to interact with SUT and cannot describe the test logic.

TTCN-3 looks like an ordinary programming language with expanded test, including dynamic testing configuration, matching mechanism, timer support, processing test decision and communication mechanism. The testing system provides the mechanism to execute TTCN-3 from two levels of the execution logic and test adapter. The TTCN-3 test-case can be independent to the SUT and the test platform through a standard interface.

III. THE DISTRIBUTED TESTING SYSTEM

The wide application of distributed technology also gradually affects the design of testing system. On one hand, in order to test the distributed software system, the testing system

has to support the distributed testing; On the other hand, the testing system which adopts distributed architecture not only make full use of limited hardware resources and human resources, improve the efficiency of executing test, reduce the cost of time and resources, but also can simulate the actual operating environment, support some special testing scenarios and produce the required test load.

The distributed execution framework defines the system structure of testing execution platform, realizes the effective management of all testing execution nodes, and provides the basic services needed to execute the distributed testing. For the choice of the distribute architecture, this paper adopts the distributed strategy which is centralized. The distributed system is shown in fig.1.

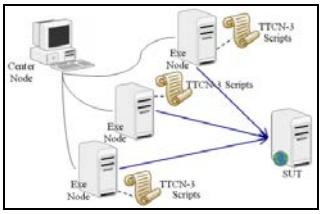

FIGURE I. THE DISTRIBUTED ENVIRONMENT.

It includes two kinds of nodes, that is Centre Node and Execution Node. The TTCN-3 testing script runs on the execution node, it describes a use-case how to test the system, including test input, test results and test execution conditions. It completely describes the process of testing the system in a given test target. The distributed testing system designed in this paper is shown in fig.2.

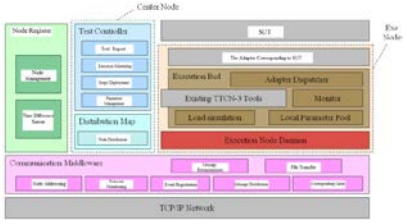

FIGURE II. THE ARCHITECTURE OF DISTRIBUTED TESTING SYSTEM.

It includes three kinds of nodes, which are Centre Node, Execution Node and Nodes Register. The interaction among them is realized by the communication middleware which transfers the asynchronous communication information based on message and is realized by the socket based on TCP(Transmission Control Protocol). The execution node communicated with the SUT through the test adapter, as to meet the needs from the SUT(System Under Test) which needs different codec rules and interactive communication modes.

Based on the transaction as a unit, the communication resources are organized and managed. In the process of running transaction, the thread to deal with the communication process needs to put the resources into the recovery device, the resources are not able to determine whether will be used in the future and are recycled when not in use. The distributed task can be completed by collaborating which involves information interaction. Therefore, in order to make the different nodes exchange data and cooperate effectively according to the requirements, we must formulate corresponding rules for interactive tasks and synchronous timing. This paper adopts the hierarchical division way to organize the testing task execution. Tasks are independent between layers, each layer doesn't need to know the mechanism to implement the next layer of it, and only need to know the interfaces between the layers. When the mechanism to implement a layer is changed, the upper and lower layers of it are not affected.

IV. THE PERFORMANCE TESTING

A. *Describe and Control the Performance Testing*

The performance testing model studied in this paper combined with the advantages of load simulation based on network characteristics and user behaviours, the performance testing scenarios based on virtual users and load indexes are presented. They provide the basis to describe and control the performance test-cases. Virtual users represent the users to use a function of the SUT, so as to simulate user behaviours in accordance with the business process in an actual operation. The testing process based on virtual users is an iterative process of concurrent behaviours. To design the performance testing needs to consider how to organize virtual users. So the virtual user organization hierarchy is proposed and shown in fig.3.

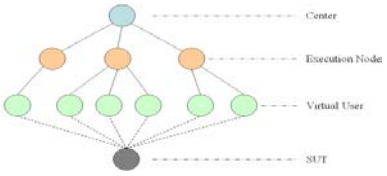

FIGURE III. THE VIRTUAL USER ORGANIZATION HIERARCHY.

The node mappings allocate the tasks between the execution nodes to make use of the execution resources in the distributed environment. Within a node, a test bed is responsible for creating and destroying the virtual users, which maintains the number of concurrent virtual users required at a level, so as to simulate the actual load. This paper introduces two new indexes and a performance index. The load model is based on above three indexes and shown in fig.4.

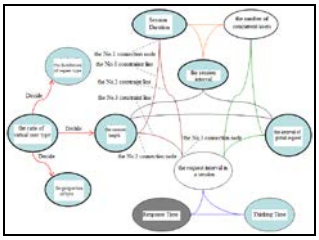

FIGURE IV. THE LOAD MODEL.

In the fig.4, each index is connected with the curve which is called as the constraint line. If each index is regarded as a circle, the two ends of each connecting line were located separately in a point on different circles. The point is called the connection point. For the set V composed of three indexes, there are three connecting points according with the indexes located in the V. If there are three constraint lines respectively

to the three connection points for the end points to form a circle, the three indexes from V constitute a constraint group.

B. The Performance Testing by TTCN-3

Through TTCN-3 language and TTCN-3 system, we can construct the performance testing framework and realize the virtual user management, load controlling, dynamic parameter management, communication connecting management, execution monitoring and track record. The performance testing framework implements the execution bed of the execution node in a distributed testing environment. The execution bed is the driver to execute TTCN-3. Its purpose is to control the execution of TTCN-3, and then implement the virtual user management, load generation, test monitoring and resource management. The relationship among the function nodes from the framework is shown in fig.5.

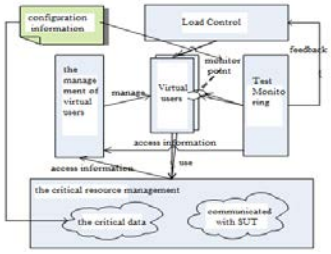

FIGURE V. THE RELATIONSHIP OF PERFORMANCE TESTING FRAMEWORK.

In order to reduce the development workload and shorten the cycle with greater efficiency, an automatic generator of TTCN-3 performance testing code is implemented in this paper. In a test cycle, the function testing precedes the performance testing, which means that a lot of available function test suites have already existed before the beginning of the performance testing, including TTCN-3 testing code, test adapter and codec. On the other hand, after 7 years of developments, TTCN-3 already has a plenty of function test suite that are developed, which are the basis of performance testing. In this paper, the TTCN-3 function test-cases are parsed by the TRex open source tool (the TTCN-3 Refactoring and Metrics Tool), then an abstract syntax tree is generated. According to the position information of the source code corresponding to the node in the syntax, the replacement node is labelled in the source code. By analysing the syntax tree, the new code is generated for the replacement corresponding to the nodes, and at last the original code is read. The characters outside the replacement are output as original. The characters inside the replacement are output as new codes to generate the performance test-cases.

V. CASE STUDIES

In this paper, the performance test-cases are generated on the basis of existing TTCN-3 function test-cases which describe the process of using an online bookstore and inspecting and deciding the behaviours of SUT. The test deployment scripts are mainly used to design the distribution scheme of test scripts in different test points and the communication between different test points and the test control centre, and the path of the TTCN-3 script file deployed in each test execution node. The load configuration describes the performance testing scenario which is composed of Phrases with differ load levels to describe the input intensity submitted by different users, including the command, task and data submitted. In this example, the test load configuration script contains 11 Phrases of the static load index to describe the ratio of different types of virtual users. Every Phrase has a different dynamic load index. The performance of SUT is focused on when the number of concurrent users are different, so the number of concurrent users and the thinking time are chosen to be controlled, the thinking time is constant in each phrase. The test results are extracted from the performance measurement. The number of concurrent users is a variable. The response time and throughout and the ratio of success are dependable variables, it will get the metric map shown in Fig.6.

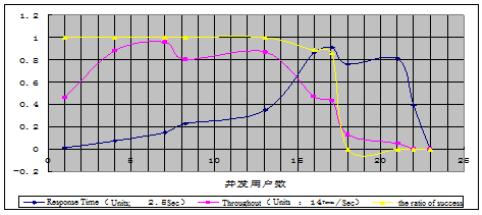

FIGURE VI. THE PERFORMANCE METRIC MAP.

In fig.6, as the number of concurrent users began to gradually increase from 1, the response time of SUT increased linearly, and the ratio of the success session retained 100%. The throughout increased at the beginning, the SUT worked normally and can respond to the load smoothly. When the number of concurrent users increased to 13, the response time significantly increased, the throughout began to decrease, the ratio of the success session also gradually decreased, then the SUT emerged the performance bottleneck. The user accesses to the online bookstore with 0.1sec thinking time, the online bookstore can support a maximum of 13 concurrent users.

VI. CONCLUSION

In this paper, the distributed execution strategy and framework are designed to support the performance testing. The method of describing and controlling the performance testing load is also proposed. And the technology to implement the load description controlling is studied, which is based on TTCN-3 system. This paper also analyses and compares the function test-cases and the performance test-cases based on TTCN-3. Then it puts forward a performance testing method based on the test suite reuse, which can automatically reconstruct the TTCN-3 function test-case and then transfer the function test-case into the performance test-case. It will reduce the development workload of testers. Finally, through the performance testing of an online bookstore, the method and process of TTCN-3 performance testing are demonstrated in a distributed environment by use of the system designed in this paper. Through the analysis of the results, the performance bottleneck of SUT is found.

REFERENCES

[1] Gábor Ziegler, György Réthy. Performance testing with TTCN-3 [R], TTCN-3 User Conference, 2010

[2] Mahnaz Shams, Diwakar Krishnamurthy, Behrouz Far. A model-based approach for testing the performance of Web Applications [J].

Proceedings of the 3rd international workshop on Software quality assurance: 54 - 61, 2011

[3] Katerina Goseva-Popstojanova, Fengbin Li, Xuan Wang, and Amit Sangle.A Contribution Towards Solving the Web Workload Puzzle [J]. Proceedings of the 2006 International Conference on Dependable Systems and Networks, 2011

[4] Raúl PeñaOrtiz, Julio Sahuquillo, Ana Pont, José A. Gil. Modeling continuous changes of the user's dynamic behavior in the WWW [J]. Proceedings of the 5th international workshop on Software and performance: 175 - 180, 2012

[5] Barford, P. and Crovella, M. Generating Representative Web Workloads for Network and Server Performance Evaluation [J]. Proceedings of the 1998 ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems:151-160, 1998

[6] Diwakar Krishnamurthy, Jerome A. Rolia, Shikharesh Majumdar. A Synthetic Workload Generation Technique for Stress Testing Session-Based Systems [J]. IEEE Transactions on Software Engineering. 32(11):868-882, 2006

[7] Helmut Neukirchen, Benjamin Zeiss, Jens Grabowski. An approach to quality engineering of TTCN-3 test specifications [J]. International Journal on Software Tools for Technology Transfer, 10(4):309-326 , 2010

[8] Ina Schieferdecker, Bernard Stepien, Axel Rennoch. PerfTTCN, a TTCN language extension for performance testing [J]. Testing of Communicating Systems, 2012

[9] Zhen Ru Dai, Jens Grabowski, Helmut Neukirchen. Timed TTCN-3 - a real-time extention for TTCN-3 [J]. Proceedings of the IFIP 14th International Conference on Testing Communicating Systems XIV: 407-424. 2012

[10] Giovanni Denaro, Andrea Polini, Wolfgang Emmerich. Early performance testing of distributed software applications [J]. ACM SIGSOFT Software Engineering Notes. 29(1):94-103, 2012

[11] Dorina Petriu. Architecture-Based Performance Analysis Applied to a Telecommunication System [J]. IEEE Transactions on Software Engineering, 26(11): 1049 - 1065, 2011