One Oriented-Web Performance Test Scheme Based on PerformanceRunner

Yue-Hua Ding[†]
School of Mathematics and Computer Science,
Wuhan Polytechnic University,Wuhan, 430048, China
E-mail: ding_mickey@126.com

Ri-Hua Xiang
Beijing Special Engineering Design and Research Institute,
Beijing, 100028, China
E-mail: xiangrihua@yahoo.com.cn

The web application performance testing is gaining wide attention due to popularization of Web application rapidly. To improve performance test efficiency, this paper chose domestic performance test tool PerformanceRunner to study, and presented one oriented-web performance test scheme based on PerformanceRunner. To illustrate PerformanceRunner how to carry out performance test, a CRM system was chosen as example to study. Test script and test scene were designed, and test result was analyzed. The test script was optimized with parameterization, transaction and rendezvous. The result shows that this scheme is an excellent choice for performance test engineer.

Keywords: Performance Test; PerformanceRunner; Generation; Execution; Analyzer.

1. Introduction

Performance is one of the most important aspects concerned with the quality of software. It indicates how well a software system or component meets its requirements for timeliness. [1]. As the use of web applications is growing, maintenance of web applications is mandatory. Web performance testing is performed to determine the behavior of web applications under workload. Because of the limitations of manual testing for web applications, automated testing tools are used to determine the performance metrics like throughput, response time, and download time of web applications. These tools provide the quality of service analysis and help to evaluate the service performance in real time network[2].

In this paper, we introduce PerformanceRunner (we call PerformanceRunner as PR below) features, and focus on how to use PR to optimize test script. It is mainly facing high-load performance test. PR can provide concurrent pressure test, monitor test and test report. PR also supports http and socket protocol, and is suitable for B/S and C/S architecture.

2. Related works

Patel M C proposes a framework for performance analysis and performance evaluation for performance driven development. They have discussed comparative analysis of various software performance testing tools and their limitations and a new approach for software performance testing. [1].

N Thakur et al. aim to provide an evaluation and comparative analysis of six web performance testing tools namely JMeter, NeoLoad, WAPT, Loadster, WebLOAD and LoadComplete. They conclude that NeoLoad is the best tool among the selected tools based on the parameters considered for evaluation of the tools[2].

Deng Xiaopeng et al. analyze the factors influencing testing for Web applications from such cases as its architecture, techniques, elements of its composition, running environment, running mechanism, design process and development process. Main aspects for web application testing in design, development, running and evolution steps are summarized. [3].

RUI Su-juan et al. presents some important measures in the progress, such as response time, throughput capacity, customer behavior[4].

ZHOU Lu-yang et al. use the UAT technology to guide and manage the whole testing flow and do the functional testing, stability testing and pressure testing research. The final result turns out that this UAT method can manage the whole testing process for Web applications in high efficient way[5].

Zhang Xiaobo sets up a software performance environment, plans and designs the scripting code so that he can guaranty the accuracy of the result effectively. He uses LoadRunner to analysis data to find system’s bottlenecks[6].

SONG Shi-tao presents an effective performance testing process model for a Web application system, based on LoadRunner, egue and PTGM performance testing models. It covers the whole process from project plan, testing policy, to result analysis and test report output. [7].

D Kelkar et al. focus on the study and evaluation of differences in responses given by tools used for performance testing. The objective is to understand the behavior of various performances testing tools and determining the accuracy of responses. They compare JMeter and LoadRunner and recommend to go ahead with HP LoadRunner as it is very stable and robust. [8].

I Arora presents an extensive Literature Survey of performance Automation Testing tools. He presents a list of most widely used performance testing tools for measuring web application performance and load stress capacity. This study has highlighted various main features of software testing tools which are essential for opting the right tool at right time[9].

3. Case Study

A customer resource management(CRM) system is selected as example to study. The CRM system is based on B/S architecture[10]. function test requirement is described, and how to implement performance test with PR is explained.

3.1. *Function Test Requirement*

The root requirements is called CRM, and sub requirements under the root requirement is constructed, and the second level sub requirements under the first sub requirement is construced. The first level Test Sub Requirement test requirement contains *system login*, *customer manage*, *contact manage*, *document manage* and *reservation manage*. *System login* requirement contains the second level test sub requirement that includes *login module*. Customer manage requirement contains the second level test sub requirement, such as *add customer, potential customer, signatory customer, terminated customer* and *customer management* [10] .

3.2. *Performance Test Requirement*

As case study, the test CRM system only focuses on two modules: *login* model and *add customer* model. There are only two conditions to satisfy the performance requirements: *Login* module can support 50 user login system at the same time within 3 seconds. *Add customer* module can support 10 add customers operation within 1 second.

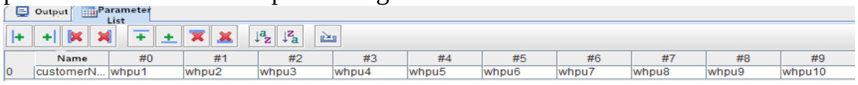
3.3. *PerformanceRunner Generation*

CRM login module and add customer module s recorded by PR generation, and the test script is optimized to enhance reusability.
(1) Login module: Firstly, login system test script is recorded. The script includes some steps, such as, open CRM login page, input username and password into text field, press the login button to enter CRM, and click quit system link. Test script needs to simulate 50 users click the login button at the same time, so *startLogin* rendezvous is added before click login button script line. In order to analyze bottleneck of *login* model, two transactions were added into test script. The part of before click login button script is encapsulated into *beforeLoginButton* transaction, and the part of after click login button script is encapsulated to *afterLoginButton* transaction. The *beforeLoginButton* transaction and *afterLoginButton* transaction execution time cost is compared,

we can analyze where the bottleneck is. There are some of the scripts displaying as below:

```
pr.startTransaction("beforeLoginButton");
pr.http_getVerb("login.asp", "http://localhost/login.asp", "");
……//some http_getVerb script
pr.endTransaction("beforeLaoginButton");
pr.startTransaction("afterLoginButton");
pr. rendezvous("startLogin");
pr.http_postVerb("submit.asp", "http://localhost/submit.asp",
"http://localhost/login.asp",     "username=test&password=test&Submit=login");
…………//some http_getVerb script
pr.endTransaction("afterLoginButton");
```

(2) Add customer module: the add customer script is recorded firstly. The scripts includes login system, click add customer link to open page, input company name into text field, and input confirm button to add customer. The script needs parameterization technology to memorize company name into data pool. New company name is parameterized as whpu1-whpu10. It is just as Fig.1 showing. So the test script can simulate add customer operation 10 times continuously. It enhances the reusability of PR test script. There is part of parameterization test script showing as below:



	Name	#0	#1	#2	#3	#4	#5	#6	#7	#8	#9
0	customerN...	whpu1	whpu2	whpu3	whpu4	whpu5	whpu6	whpu7	whpu8	whpu9	whpu10

Fig.1. Input Data Parameterization of PerformanceRunner

```
for(ParameterData pd : pr.getParameterDataList("Action.xls").subList(0, 9)){
pr.http_getVerb("main.asp_2", "http://localhost/main.asp", "http://localhost/submit.asp");
……//http_getVerb script
pr.http_postVerb("Client_Save.asp", "http://localhost/Client/Client_Save.asp", "http://localhost/Clien
t/Client_Add.asp", "Client_Company="+pd.getFrom("customerName")+"&Client_Class=A&Clie
nt_beijing=1&Client_shangshi=1&Client_ziben=&Client_xiaoshou=&Client_guimu=&Client_ceshi
ren=&Client_goumei=1&Client_ruanjian=&Client_Http=&Client_youbian=&Client_Address=&Cli
ent_zhuyingchangping=&Client_zhuyaofuwu=&Client_anlihangye=&Client_anlihangjuli=&Client_
jiagoukoubei=&Client_qingkuang=&Client_Text=&Client_Follow=N&Client_sales_1=test&Client
_sales=11&Client_Date=2016/7/22&button2=Confirm");
……//http_getVerb script
pr.http_getVerb("pic02.gif_2", "http://localhost/images/pic02.gif", "http://localhost/Login.asp");
}
```

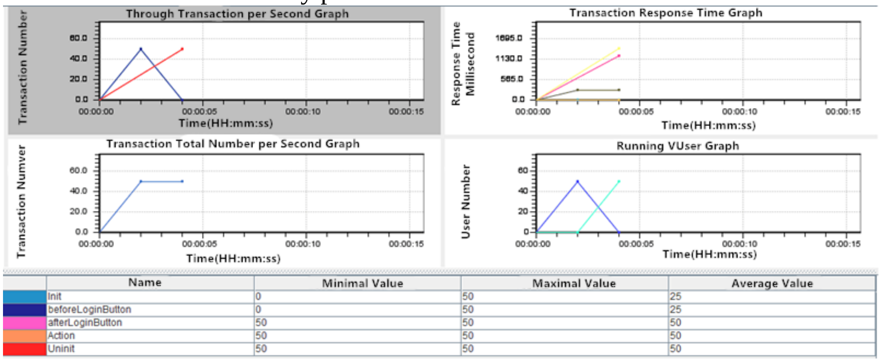
3.4. *PerformanceRunner Execution*

PerformanceRunner execution is the place where PR execute performance test. Performance test execution is started, and different performance test requirement is simulated. We will introduce how to implement login module and add customer module. After finish scene design, scene start to execute and the result is showed in execution in real time.

(1) *login* module scene design: new scene *CRMLogin* first. After define 50 virtual user, configure launch them at the same time. It won’t end until all the 50 virtual login complete. Fifty virtual users launching within one second is

simulated. Fig.2 shows *login* module execution result. From Fig.2 we can conclude that *login* module satisfy the performance requirement. It is successful to launch 50 virtual users within 3 seconds.

(2) Add customer module design: new scene *CRMAddCustomer* first, and configure only one user. One virtual user within 3 seconds is started. The virtual user will execute add customer ten times. From observing running graphics, add customer module also satisfy performance demands.

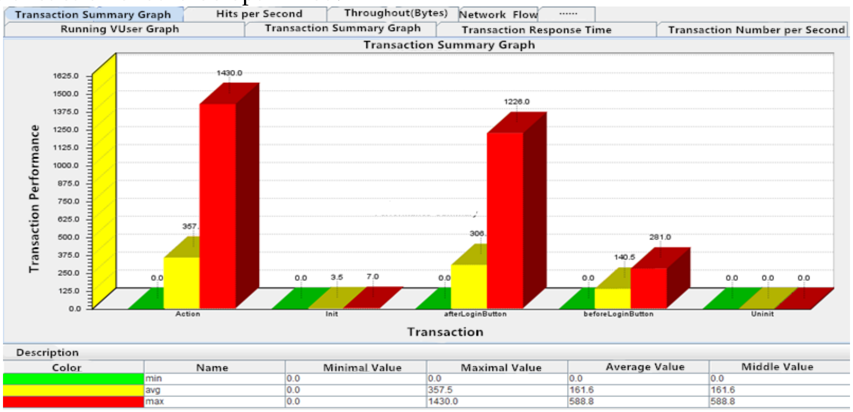


	Name	Minimal Value	Maximal Value	Average Value
🟦	Init	0	50	25
🟦	beforeLoginButton	0	50	25
🟪	afterLoginButton	50	50	50
🟧	Action	50	50	50
🟥	Uninit	50	50	50

Fig. 2. Scene Execution Monitor Graph for CRM *Login* Module

3.5. *PerformanceRunner Analyzer*

PR analyzer can make many reports to support test engineer write their test report. PR can provide VUser graph, transaction graph, transaction response time graph, hits number per second graph and so on. *Login* module transaction performance summary graph is shown as an example. It is found out from Fig.3 that *beforeLoginButton* transaction performance is better than *afterLoginButton* transaction performance. If the *login* module performance is bad, the worse transaction should be optimized.



Color	Name	Minimal Value	Maximal Value	Average Value	Middle Value
🟩	min	0.0	0.0	0.0	0.0
🟨	avg	0.0	357.5	161.6	161.6
🟥	max	0.0	1430.0	588.8	588.8

Fig. 3. Comparison between BeforeLoginButton Transaction and *AfterLoginButton* Transaction

4. Conclusion

Due to software development scale getting larger, performance problem becomes more serious than before. We present one performance test scheme based on PerformanceRunner to enhance performance test efficiency. We use generation, execution and analyzer to help test engineer to find bottleneck of system, and then enhance system reliability.

Acknowledgments

This research was supported by the Wuhan polytechnic excellent course.

References

1. Patel M C, Gulati R. *Software Performance Testing Tools – A Comparative Analysis*[J]. World Development, 2012, 14(4):489-502.
2. N Thakur, KL Bansal, *Comparative Study of Web Performance Testing Tools*, International Journal of Advanced Research in Computer Science and Software Engineering ,Volume 5, Issue 5, May 2015
3. Deng Xiaopeng, Xing Chunxia, and Cai Lianhong. *Progress in Testing for Web Applications*[J]. Journal of Computer Research and Development, 2007, 44(08):1273-1283.
4. RUI Su-juan, DING Xiao-ming. *Progresss in Web Applications Performance Testing* [J]. COMPUTER SCIENCE, 2006, 33(08):278-280.
5. ZHOU Lu-yang, JU Shu-CUl+,XU Xiang. *Test Research of Network Application Program Based on Web*[J]. Journal of Anhui Agri.Sci., 2015(27):382-384.
6. Zhang Xiaobo. *The Application of the Software Performance Test in the Bill Inquiry System* [D]. Guangdong University of Technology, 2015.
7. SONG Shi—tao. P*erformance test process of a Web application system and its application*[J]. SHANDONG SCIENCE, 2015, 28(04):94-100.
8. *Analysis and Comparison of Performance Testing Tools*, D Kelkar,K Kandalgaonkar, International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), Volume 4 Issue 5, May 2015
9. I Arora , *A Brief Survey on Web Application Performance Testing Tools Literature Review*, International Journal of Latest Trends in Engineering and Technology (IJLTET). Vol. 5 Issue 3 May 2015.367-375
10. Yue-hua Ding, Ri-hua Xiang, *One Automated Test Scheme Based on AutoRunner and TestCenter*, 2nd International Conference on Wireless Communication and Sensor Network (WCSN 2015)[C],951-959