

STeP-IN SUMMIT 2007

International Conference On Software Testing

User Behavior and Performance Perception Analysis - An Approach for Realistic Performance Engineering

by
RAHUL VERMA

APPLABS TECHNOLOGIES, Hyderabad

rahul.verma@applabs.com

Copyright: STeP-IN Forum and Quality Solutions for Information Technology Pvt. Ltd.

Published with permission for restricted use in STeP-IN SUMMIT 2007 in agreement with full copyrights from owner(s) / author(s) of material. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise without the prior consent of the owner(s) / author(s). This edition is manufactured in India and is authorized for distribution only during STeP-IN SUMMIT 2007 as per the applicable conditions.

Practices Experience Knowledge Automation

Produced By

STeP-IN
Forum

www.stepinforum.org

Hosted By



www.qsitglobal.com

ABSTRACT

User Behavior and Performance Perception Analysis (UBPPA) is an approach for carrying out the performance engineering activities in a realistic manner. This approach breaks the myth of a so-called “general performance testing” approach or capacity sizing guidelines that work for all applications. Instead, it stresses upon the fact that the performance of an application is perceived by different users in different manner. It also discusses the importance of analyzing the needs and expectations of the target users for the application. With this approach, the stakeholders can be sure that they do not end up over-sizing or under-sizing their application in terms of hardware or configuration settings on their servers in urge for increasing an unneeded performance factor. UBPPA addresses the performance needs of the application from end users’ perspective.

The white paper will be logically divided into following sections:

1. User Behavior and Performance Perception Analysis (UBPPA) – An Introduction

This section will address, in general, how performance engineering activities are carried out and how UBPPA can strike a difference.

2. User Behavior – Usage pattern, user mix and think time

This section will address analysis of user behavior. By understanding user behavior, an optimal workload distribution pattern can be crafted, thereby facilitating a realistic performance test. It also, helps us to decide an optimum user think time.

3. User Oriented Performance (UOP)

This section will address the concept of user oriented performance, the factors that govern it and how it might impact the stakeholders.

4. Usability and Responsiveness –Virtual Increased Performance

This section will address the concept of usability and responsiveness, the factors that govern it and how it might impact the stakeholders.

5. Performance Testing with UBPPA

This section will cover running a successful and realistic performance test by applying UBPPA.

6. Performance Tuning and Capacity Planning with UBPPA

This section will cover the importance of UBPPA in performance tuning and capacity planning of the application.

7. Conclusion

1.0 User Behavior and Performance Perception Analysis (UBPPA) – An Introduction

Performance Engineering has grown from an occasionally discussed term in the recent past, to a full-fledged, self-sufficient, focused profession. It was introduced as testing an application at the end of development and functional testing cycle and now is being seen as essential during the design phase itself. Many Performance experts around the globe have done a lot to imbibe maturity into performance engineering approaches. Putting it all together, things have changed very fast for performance engineering.

There has been enough discussion on designing an application from end users' perspective, doing performance testing keeping real users in mind and also about perceived performance. All this can be found in the form of articles on the web, some of which I have mentioned in the *References* section in this paper. This paper introduces all these aspects put together as a separate, focused science - **User Behavior and Performance Perception Analysis** (referred to as **UBPPA** here onwards). UBPPA is an attempt to change the way we look at performance engineering.

UBPPA is all about '*Who will use it and how they will use it*' being the governing factor for designing and testing an application. UBPPA, as depicted in Fig 1, is a strategic mix of the following **Three U's**:

1. User Behavior
2. User Oriented Performance
3. Usability and Responsiveness

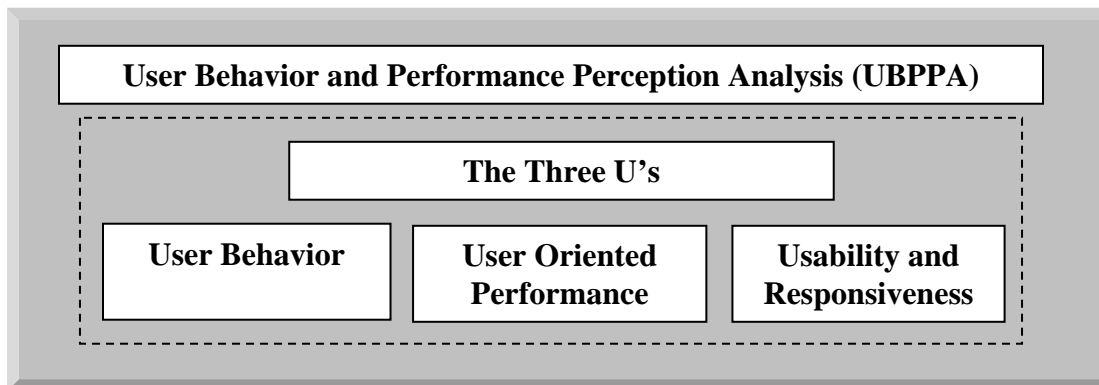


Fig 1.0

The next three sections will discuss one by one the three U's and the following sections will discuss how UBPPA as a science can be applied in effective performance testing, performance tuning and capacity planning.

2.0 User Behavior – Usage patterns, user mix and think time

In performance engineering terms, user behavior is nothing but the user distribution pattern and his knowledge of the application/platform. It is more than simply talking about the maximum user load. Talking about user behavior includes considering the following questions/suggestions about your application:

2.1 Usage patterns

1. **Type of application:** Is it an internet application or intranet application?
2. **Maximum number of users:** In a day how many users do you expect to use your application?
3. **Usage pattern variation:** For internet application, in a day, what are the peak and dull hours? i.e. for what part of the day you expect to see more usage of the application? For an existing application, server logs will be of good guidance for understanding the usage patterns. For an application yet to be launched there has to be lot of market and statistical analysis to be done to assess the possible usage.
4. **Consistency of Login:** For intranet applications, what is the frequency of login? To simplify, is it an application like a teller application, where the user is logged in for almost whole day and consistent requests hit the server(s)? Or is it an intranet application, where you see only intermittent logins or usage?

The above are only a few points to give you a direction regarding the user behavior. The questions and their numbers will vary depending upon the type of application. The graphs shown in this figure show you a glimpse of some usage patterns.

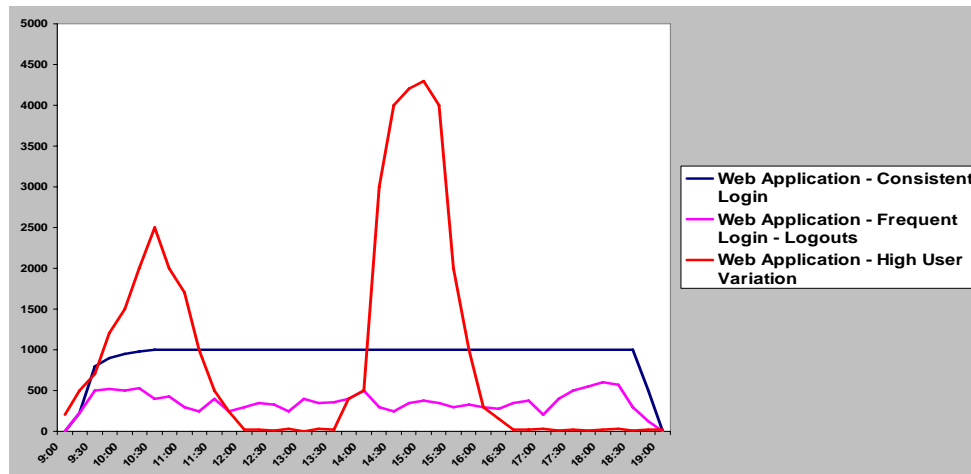


Fig 2.0

So, a web application which will be used by 1000 users with consistent logins should be quite different from one designed for frequent login logouts. Also, for an application with high variations in load, the design as well as the testing approach has to be different.

2.2 User Categorization based on privileges

1. How do you categorize your users in terms of privileges? For example, there might be say only 2% users with administrator privileges, another 10% with partial administrator privileges for certain options and the rest normal users.
2. For each category of users, one needs to separately consider the points discussed in the section 2.1 Usage patterns.

2.3 User Categorization based on users' knowledge of application

1. How do you categorize your users on the basis of knowledge of application? They might be categorized into new, average and fast users.
2. New users will take more time to fill the forms; they will be sending requests at a lesser frequency than others. This frequency will be more for average users and highest for fast users
3. Assessing the percentage mix of the users based on expertise of the application in addition to considerations to the complexity of form data, location of links etc for individual transactions helps one to come up with realistic user think times, i.e. the think time which a new, average or fast user, on an average, takes to think (or to fill forms) between sending two consecutive requests.

Other user categorization might be based on different bandwidths, browsers, hardware configurations and so on so forth. Section 5.0 onwards, I will discuss, how the user behavior analysis helps in performance engineering activities.

3.0 User Oriented Performance (UOP)

UBPPA is not just about '*Performance*', it's about '*User Perceived Performance*', and there is lot of difference between the two. End user does not bother (in fact he would not even know) how many servers you have put, how large is your bandwidth, implementation of load balancing or the routing algorithms. If the end response is not as per his expectation, it is 'SLOW' for him. A normal misconception is that when an application sends the response in 2 seconds, instead of 5 seconds, it is faster. Logically, it is absolutely correct. But when we map it to end users' experience, the answers might be significantly different from what we expected.

I came across various discussions on User Perceived performance on the web. I have purposely split the concept as comprising of two different concepts namely:

- a) **User Oriented Performance** – Increases Real performance, by focusing on what performance means to the target user. Includes performance tuning and sizing of the application for the identified areas.
- b) **Usability and Responsiveness** – Increases Virtual performance. It just *appears to be fast* to the end user. Mainly deals with usability aspects of the application with respect to GUI design, batch processing etc. Refer next section for more details on this.

User oriented performance is directly dependent on the following:

1. Purpose of the application as a whole has direct impact on the performance which the user expects.
2. For a web application, performance expectation of the user also depends on the fact whether he is paying for the service (a registered user) or is a free user. Normally, paid sites should maintain high performance standards. A normal site should differentiate between the performance experience of a registered user and unregistered user, if registration costs.
3. Performance expectation of an end user is also governed by the fact by the performance of other similar applications in the market.
4. Certain business flows of the application attract more users and will see more traffic. E.g. more users will surf a shopping site for looking for products, than for buying it. These flows will need more attention for performance. For buying scenario in this case, fewer resources will suffice, not for poor performance, but because of the fact, this flow will see less user load. This analysis will help you come up with **Transaction mix**. Transaction mix combined with user patterns is an essential input for performance test scripts for simulating user traffic.
5. To the end user, a transaction which is a bit slower will appear faster than the one which gets HTTP error codes in response and for which he has to again send the request. So, handling concurrent users is not only important for scalability but has direct impact on the end user experience with respect to response times as well.

Scott Barber in his series '*User Experience Not Metrics*' stresses on an important aspect 'Abandonment of website' by users. Users abandon the site in case of slow responses. This aspect is rarely taken care off, and should soon be introduced in the tools as killing the thread based on response times, rather than continuing the iterations. This might as well be taken care by developing customized tools. User abandonment brings down the load on the application and gives realistic calculations. This might as well be related to calculations for business loss based on how many users abandoned the site.

4.0 Usability and Responsiveness –Virtual Increased Performance (VIP)

Everything that glitters is not gold. But if it's not gold, and still glitters, it still makes sense. This is the whole concept of Usability and Responsiveness. Once implemented, it makes the application '*appear to be performing better*'. I will use the term Virtual Increased Performance for usability and Responsiveness put together.

Virtual Increased Performance can be imbibed in the application through the following:

1. Good GUI design from usability perspective
2. Employ Pareto's 80-20 Principle while placing links on the website
3. Keep the user experience interactive. Do not let the user wait until the operation is completed. Let him know about the progress of the operation at intermittent intervals.
4. If the operation is going to take more than 1 minute, plan it to be a background process, so that the user can work on other things in the

application. He can be intimidated asynchronously about the completion of the operation.

Although the above do not require any necessary performance tuning and sizing, and is more inclined towards the GUI design of the application, but still finds relevance in performance engineering. And since, VIP is about users, it's has to be considered under UBPPA.

5.0 Performance Testing with UBPPA

Having discussed about UBPPA, in whatever dimensions this white paper allows, I will now talk about how UBPPA can be applied to performance testing projects. In a performance testing project, a lot of organizations follow the Performance testing methodology suggested by Scott Barber and there are many others including AppLabs, who have worked further on the same and have come up with customized performance testing approaches. This paper is not about discussing which approach is better; rather it is going to point to those sections in performance testing, which undoubtedly will be common to any methodology for performance testing.

I have worked with different commercial as well as open source tools for performance testing. The following points are achieved either through the GUI settings of the application or mainly through scripting done in the tool-specific language. The same can as well be extended to your own 'personalized testing tools' in language of your choice like Perl, Python, Java etc.

- 1. Deciding User mix and transaction mix:** Some of the essential initial inputs for automated performance testing of an application with simulation tools are user mix, and transaction mix. These can be got from UBPPA, as discussed in the User Behavior and User Oriented performance sections. A typical user mix might look like this:

User Type	User %	Transactions	Transaction %
Admin	5	Admin1	10
		Admin2	90
Normal	95	Normal1	20
		Normal2	30
		Normal3	50

- 2. Ramp-up pattern:** This describes how the simulation tool will spawn threads to simulate realistic user traffic. This also can be got with UBPPA, as discussed in 2.1 Usage patterns, and then applied programmatically through Tool. This will help the tool to test the application for real time usage rather than blindly putting a step-up ramp-up or linear ramp-up.
- 3. User think time (Sleep time):** This is again an essential part of the performance testing. User think time is the time that a simulation tool should wait between sending consecutive requests within a thread. This again helps in user modeling, as discussed in the User Behavior section. It can enhance the above mix table as follows:

User	User	User %	User Think	Transactions	Transaction
------	------	--------	------------	--------------	-------------

Type	Speed		Time (secs)		%
Admin	Fast	5	2	Admin1	10
				Admin2	90
Normal	New	15	5	Normal1	20
				Normal2	30
				Normal3	50
	Average	60	3	Normal1	20
				Normal2	30
				Normal3	50
	Fast	20	2	Normal1	20
				Normal2	30
				Normal3	50

The above is a simple example and numbers may vary from application to application. On the same lines, much complex mixes can be crafted based on real data and then implemented in the simulation tool. Scott Barber suggests '*User Community Modeling Language*' (UCML) for graphical representation of the usage distribution patterns, business flows and transaction mixes, instead of tabular representations.

4. Session Management (Consistent logins): This is one of the most neglected fields in performance testing projects. Many a times people craft their scripts in such a way that session is being maintained unnecessarily (doing transactions in the same login for whole duration of the test), and sometimes they do not do it when actually it was required. For more clarity on this please look back at Fig 2.0, which depicts the difference between consistent logins and otherwise. So, if one is not dealing with sessions in the way real time usage expects, one is doing a wrong performance test.

The session management can again be done on a mix basis, marking what percentage of what category of users will be logged in consistently. The same can be programmatically implemented in the scripts.

6.0 Performance Tuning and Capacity Planning with UBPPA

Performance Tuning and Capacity Planning are vast topics and there is no general way to talk about both. They vary from platform to platform, application to application. But one thing I find common to performance tuning and sizing of any application, and that one thing is implementation of UBPPA.

Following are some points based on UBPPA, which if considered can result in effective and efficient performance engineering:

- 1. Tune and size the application for the users:** I have purposely kept this heading so simple. You might say "Man! That's what we always do!!". If that's true, you are doing what this paper is meant for! The question is that do we all really tune the application for the user or for what we thought will be the best for the user, without in-depth analysis. First try to assess which parts of the application will be used more frequently and by more number of users and then tune or design those accordingly. Allocate more resources to those parts.

2. **Configuring the application:** There will generally be a conflict in configuring the application for lowest response times and for highest number of concurrent users. With UBPPA, once you have done the user behavior and the user oriented performance analysis, you will know whether your users need the quickest responses and can bear a non-frequent server busy errors, or the errors drive them crazy and they can rather compromise over somewhat slower response times. Tune and size the application accordingly. It might not be as easy as said above and different combinations of prioritizing the transactions and concurrency will have to be tried.
3. **Configuring for handling long sessions:** In applications, where consistent logins will take place, the timeouts for sessions in case of end user inactivity have to be kept accordingly. In such an application, session management and concurrency are more important than minimizing the response times.
4. **Configuring for frequent login-logouts:** In most of the applications, login and logout are heavy operations because of being database intensive, security issues and session creation/destruction. In the applications which see frequent login and logout, a user might be offended with slow logins/logout. So, for such applications, minimizing the response times becomes more important than concurrency and session management.
5. **Differentiating registered/non-registered users:** Where money is involved, users become more demanding. Many applications concentrate more on providing extra benefits in the form of extra downloads or privileged accesses. While maintaining decent performance for non-registered users, application design should be done in such a way that the registered user can have a better experience in terms of application performance as well. At least, being a performance engineer, that makes more sense to me.

7.0 Conclusion

UBPPA is a separate science altogether which when applied in understanding users, their needs and expectations, can help us build applications that will be high performing in end-users' perspective. UBPPA as a serious effort can give an additional dimension to the existing performance engineering approaches.

References

1. *User Experience Not Metrics* – Scott Barber – www.perftestplus.com
This series of articles by Scott Barber is the main source of inspiration for UBPPA. The series talks about real user modeling and UCML – User Community Modeling language.
2. *Beyond Performance Testing* – Scott Barber – www.perftestplus.com
This series of articles by Scott Barber is built on the User Experience Not Metrics and discusses on important stuff like “How fast is enough” for an application.
3. *Integrated Approach to Web Performance Testing: A Practical Guide*
– B.M. Subraya, Infosys Technologies – IRM Press

About the Author



**RAHUL
VERMA**

Rahul Verma is currently working with **AppLabs Technologies**, Hyderabad in the capacity of a Team Lead in **Performance and Certification Testing Services Unit**. With more than four years of industrial experience, he is well equipped with software testing concepts and various open-source/commercial tools related to performance testing and functional testing for web applications, web services and thick client applications. He has presented several technical and non-technical papers so far.