



Executive White Paper Using Load Testing to meet Your SLA

By Eran Witkon, Vice President Product & Development, RadView Software

This white paper explains how systems behave under load, how to implement load testing for optimal results and how to use goal-oriented testing to verify compliance with load requirements and meet your SLA.

August 2007

Table of Contents

System Behavior under Load.....	3
The 3-Step Load Testing Process.....	4
Goal-Oriented Testing	5
Response Time.....	6
Other Goals to Consider	6
Goal-Oriented Testing as a Method to Meet your SLA	6
Finding Your System’s Degradation Point	6
Conclusions.....	8

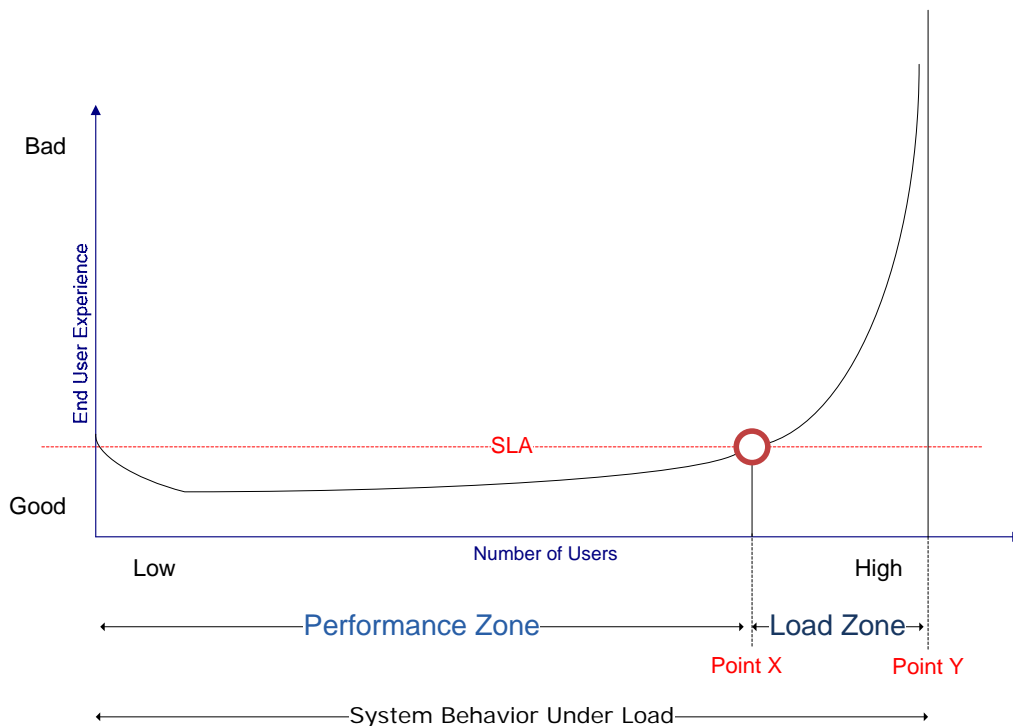
System Behavior under Load

The main purpose of the testing process is to measure an application's performance against the key requirements demanded of it. The performance of Internet applications in particular can be determined by a combination of:

- End user experience – usually measured by the end-to-end response time for common application tasks
- System resource utilization - usually measured by a set of performance counters such as CPU utilization

Client/Server Systems and Internet Applications behave similarly in response to an increase in users. As shown in Graph 1, the first few users usually improve the response time and therefore the end user experience¹ (cache utilization as data is loaded into the system memory and the system is warmed-up). From that point forward however, we see a slow, linear degradation in response time as more and more users are added to the system. When we reach a **point X** the response time begins to degrade exponentially until the system breaks at **point Y**.

Graph 1



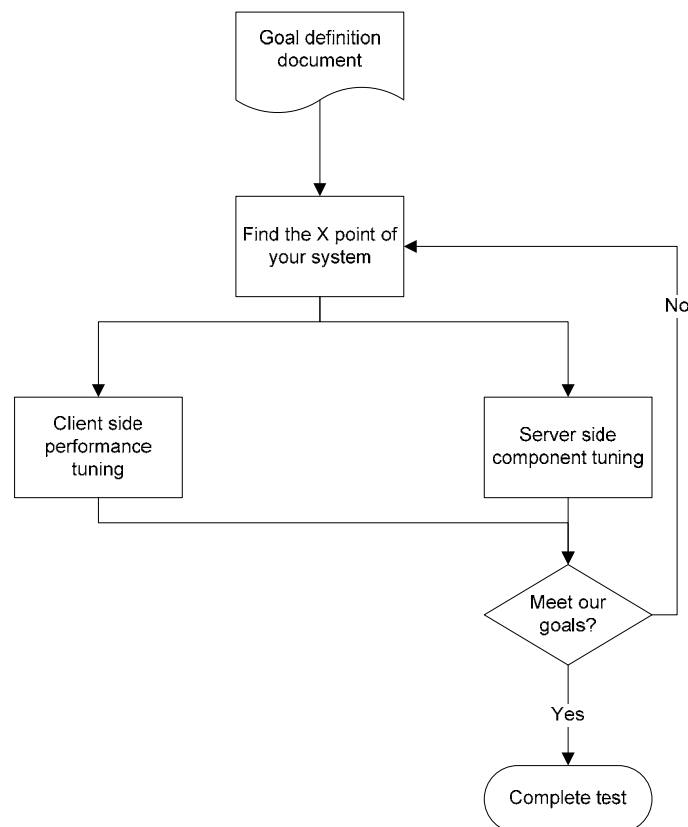
¹ We will use the term response time throughout the document as an accurate representation of end-user experience.

Based on this, we can divide system behavior into two parts:

- **Performance tuning zone** - the number of users is below point X and response time is mostly affected by the system architecture in general and the client-side architecture in particular².
- **Load zone** - the number of users is greater than X, the system under test starts to run out of resources, and response time becomes mostly affected by the server side. If we continue to increase the number of users beyond the X point, we eventually reach point Y, where the system completely breaks and cannot provide service.

The 3-Step Load Testing Process

With this in mind we can conceptually divide the load testing process into three steps as shown in the following diagram:



² Per Yahoo performance group, in this zone, client architecture is responsible for 80% of end user performance. See [Yahoo YSlow](#) for more info.

The first step is to analyze the system behavior and lay out the time graph as shown in Graph 1. The outcome of this step is the identification of points X and Y. From a business perspective, we can now answer the following question: **How many users can our system serve until the end user experience becomes unacceptable?**

The second step is to drill into the performance zone and run client side performance testing. This process will help validate and improve end user experience. WebLOAD probing client statistics in combination with tools such as firebug (<http://www.getfirebug.com/>) and YSlow (<http://developer.yahoo.com/yslow/>) can help perform this tuning process.

The third step involves advanced load testing aimed to improve system load behavior by pushing point X further to the right and/or improve system HW resource utilization to get to point X with fewer resources (cost reduction).

Once we reach beyond point X where the response time begins to degrade exponentially, we have to find the constrained resource on the server and tune the component that is using it. WebLOAD client and server side measurement correlation help us do this. The same process can be repeated in stability tests, where the system is run with the same goals for a long period of time.

The importance of identifying points X and Y cannot be stressed enough as these clearly show if the system meets the defined requirements and enables decision-makers to make a fast GO/NO-GO decision.

Goal-Oriented Testing

The first rule of load testing is “to know when to stop”. This means that defining your requirements or goals is your first action item and stopping the test process, when you reach these goals, is the second one.

Load testing without clear goals is like writing software without requirements - it might work but you don't know what the end result will look like.

When we talk about goals in the context of load testing, we refer to a number of factors including response time, CPU and memory utilization, etc. From now on we will call these measurement goals. Each measurement goal is associated with a threshold, which identifies the maximum capability of each.

Response Time

The most obvious goal from the user-experience perspective is response time.

Response time: in WebLOAD, response time includes only the server side time - Time to first byte + Receive Time - Process Time. To measure the overall response time, we use a different measurement named **Hit time**.
Hit Time in WebLOAD, is the time it takes to complete a successful HTTP request, in seconds. (Each request for each gif, jpeg, html file, etc. is a single hit.) The time of a hit is the sum of the Connect Time, Send Time, Response Time, and Process Time.

For most complex systems, 3-5 seconds represents a good average response time while 5-7 seconds is the maximum response time. Obviously the simpler your system, the faster it can and should perform.

Response time should be our most important goal. In a recent interview with Marissa Mayer, Vice President, Search Products & User Experience at Google, Ms Mayer noted that users are not aware of how **impatient** they are with regards to response time. She shared that users reacted badly when Google changed their algorithm and increased response time by 0.5 seconds.

Other Goals to Consider

If we want to ensure that our system is stable and the responses are sustainable over time, we need to make sure the CPU utilization does not exceed 85%. Running a system with constant load above 85% will make it unstable.

We can set similar thresholds for other resources, such as memory. Shortage in memory will crash a system faster than any other resource. Monitoring counters such as Page Faults/sec can tell us how many times we are paging to the disk to find our data in the memory. Other measurements that can be monitored are network utilization and disk activity.

Goal-Oriented Testing as a Method to Meet your SLA

Few application development teams have the privilege of knowing for sure how many users will access their application. They can only estimate the number of potential users. What we do know for certain is the quality of service that we want our users to receive. Goal-oriented testing is ideal for this situation. **We use the SLA requirements as our goals.**

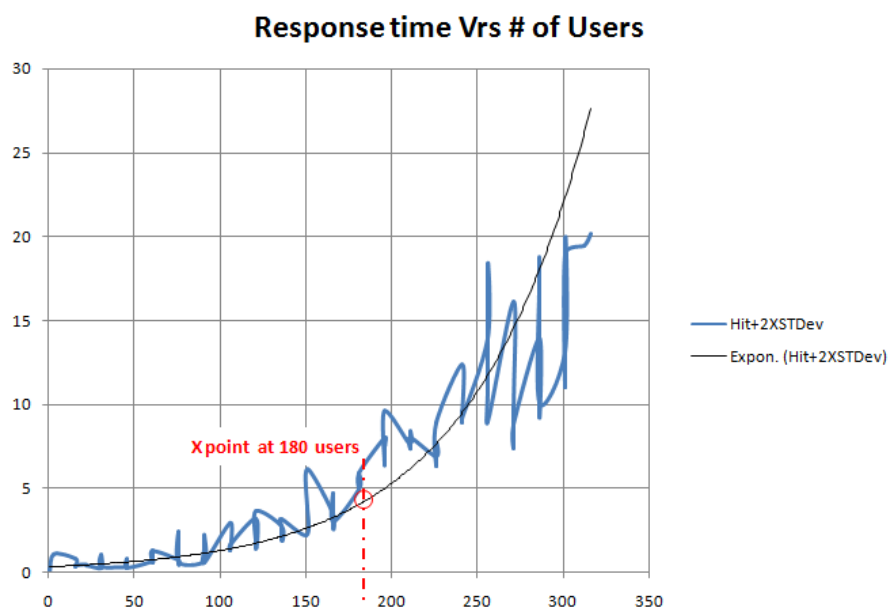
Finding Your System's Degradation Point

So far, we have explained the need for goal-oriented testing and reviewed some of the common goals we should set for our load testing. But how can we configure our test? There are two ways to do this:

- **Trial and error** - the typical process is based on running several iterations (trials) of a load test, each time with a larger number of users, while monitoring the system resources. This method tries to answer the question “What will be the system response time, if I run N number of users?” The number of users remains constant throughout each separate test. We run a few test iterations in order to find the approximate number of users that breaks the system.
- **Cruise Control** – this patented WebLOAD capability enables the gradual increase of users during a single load session, while monitoring the goal measurements until the defined thresholds are passed. Instead of guessing your load size you can efficiently and systematically run a single test run with Cruise Control and find the maximum load under which the system performs as desired. Now our questions can be more sophisticated without adding significant time and resources to the testing process. We can easily find out: **“How many users can I run on my system, while preserving a response time of less than or equal to 5 seconds?”** or: **“How many users can I run on my system while preserving a response time of less than or equal to 5 seconds AND server side CPU utilization of less than 85%?”** With WebLOAD we can add more and more parameters to our question and still receive an accurate response, quickly.

Graph 2 shows how response time grows as more users were added to the test; the black line is an exponential trend line.

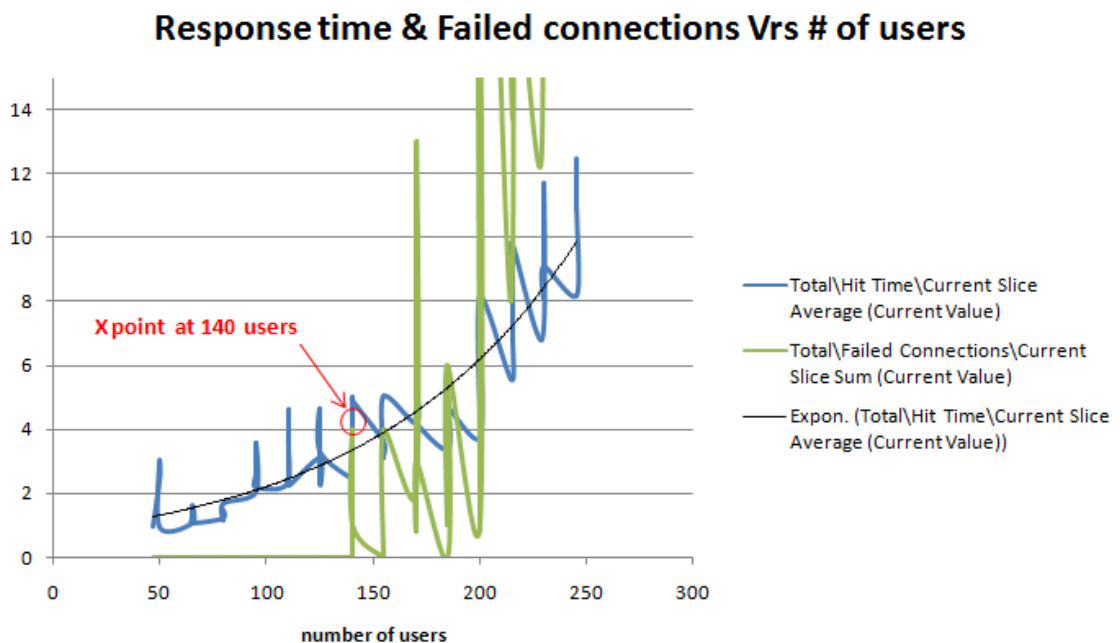
Graph 2-Finding the X point in real life



In this test we see that the SLA goals were breached at 180 users. In this case (not shown in the graph), maxing out the CPU to 100%.

Graph 3 shows another example of finding point X in a test run using cruise control and measuring both response time and the number of failed connections. As shown below, until 140 users the system did not have any failed connection. Above and beyond 140 users we started see these failures and as the load on the server increased the number of failed connections increased as well. The black line shows the exponential trend line of the response time.

Graph 3-Finding point X using failed connection



Conclusions

End user experience is a key criterion for the adoption and use of your application and is reflected in your SLAs. End user experience becomes negatively affected as greater number of users slow down your application's response time. This problem is relatively easy to manage if you know the limits of your application – point X, where performance deterioration becomes exponential, and point Y, where your system breaks under load.

Goal-oriented testing enables you to focus your tests on your SLA goals and to find points X and Y relative to your SLA. WebLOAD offers a unique capability to identify points X and Y accurately and cost-effectively, allowing decision-makers to quickly reach a well-grounded go/no-go decision regarding system readiness.

Contact Information:

North America	RadView Software, Inc. 111 Deerwood Road, Suite: 200 San Ramon, California 94583 Phone: +925-831-4808 Fax: +925-831-4807 Toll Free: +1-888-RADVIEW
United Kingdom	RadView Software (UK) 90 Long Acre Covent Garden London WC2E 9RZ Phone: +44-207-716-5840 Fax: +44 207-716-5740
Other Countries	RadView Software Ltd. 14 Hamelacha Street Rosh Haayin 48091, Israel Phone: +972-3-915-7060 Fax: +972-3-915-7683

RadView corporate website: www.radview.com
WebLOAD community website: www.webload.org