Baseline Testing

By Johann du Plessis

December 2009

Baseline Testing

Introduction

Baseline tests form a very important part of the performance test methodology I follow. If done properly, 85% of performance problems can be identified and solved when proper baseline tests are done. Recent experiences on three different projects show this figure to be closer to 90%. This really has me very excited about the whole idea and implementation of baseline testing.

What is the baseline testing referred to here? Why and how are baseline tests executed? What is in place during these tests? How do baseline tests differ from other performance tests? In this article I'll answer these and some other questions about baseline testing.

Why Baseline Tests?

The main advantage of running proper baseline tests is the time that is saved by this. Performance testing is a complex process and most often enough time is not available for proper test cycles. The time problem often leads to less baseline tests being done, but over time I've learnt that this is not the place to try and save time. In fact, baseline testing is where time can be saved.

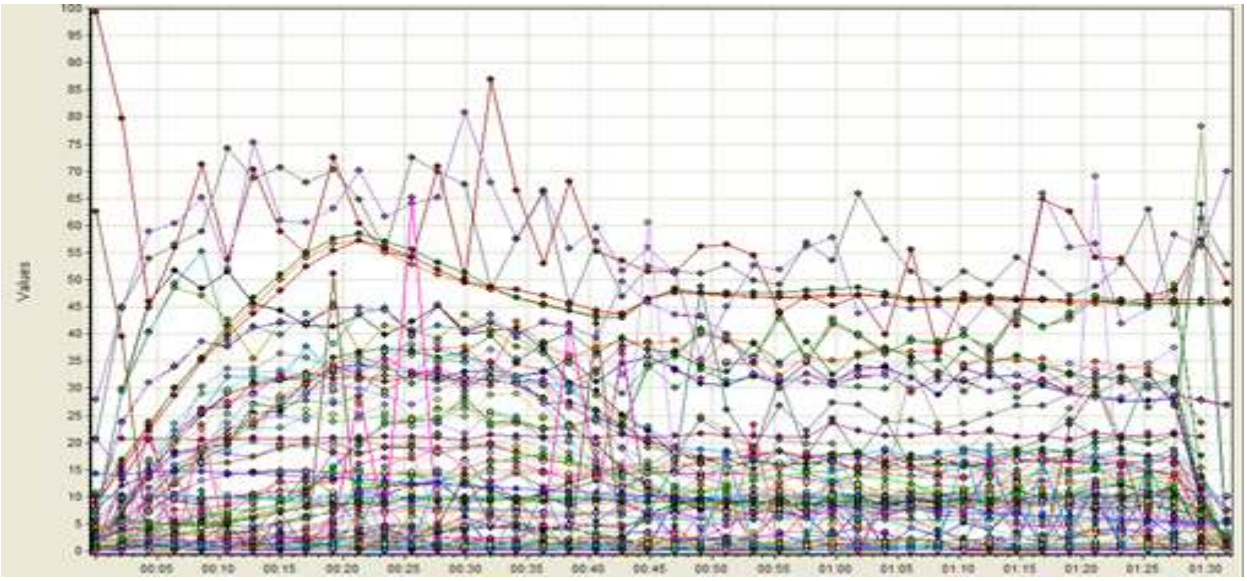
A performance test project should at least include baseline and load tests. If possible a stress test and some volume and soak (endurance) testing should also be included. There should be a clear understanding of each test type. The table below describes each type of performance testing with the terminology I use in my methodology.

Performance Testing	
Type of testing	**Description**
Baseline	Done for each script with 1, 2, 5, 10, 20 and 50 users to determine baselines for mainly response times.
Load	Test system with multiple users to determine performance under load. The number of users is usually specified by the performance test requirements.
Stress	Load the system to its breakpoint. This is to determine the system break point or threshold. How the system breaks and recovers should also be monitored.
Soak (Endurance)	Testing a system under load for an extended period of time to establish stability and behavior under sustained use.
Volume	Testing a system with a certain amount of data. Usually includes high data and throughput volumes.

When I started performance testing 9 years ago I learnt about and was trained with the term "load testing". Sometimes people referred to or used the term "stress testing" and I was told not to worry, it's the same thing. Today people still use mainly those two terms. The focus is definitely still on load testing and

there's nothing wrong with that. The end goal should always be a system that performs well under load. The way we get there is what matters.

Preparing for and executing a load test can be a tedious task. A lot of project time is spent on the preparation for load test execution. Data setup and management is one of the main areas that take a lot of the tester's time. Scripting, data preparation, linking data to scripts, scenario planning and configuration, test environment preparation and monitoring all form part of the preparation before a load test can be executed. Once everything is in place the test is started, but it seldom happens that the first load test is a success. Most of the time numerous errors occur and some users will pass while others fail until all users eventually fail. The result is usually a big mess within a very short time and data that took hours to prepare will have to be prepared again. Monitoring also becomes a nightmare with different errors experienced in different places. The result of this is very busy graphs that are difficult to interpret as can be seen on the example below.



The problem here is that a lot of time is needed before the next test can be run and it can take a lot of time before something useful can be extracted from the test results. More useful results are needed in a shorter space of time and that's what baseline testing delivers.

Useful Results Quickly

Having useful results to show quickly is a bonus on any project. As mentioned before the preparation for a load test can take a long time. During this time people will be waiting impatiently for the first test results. If the results aren't very useful or take a long time to analyze, the situation can become difficult to manage. On the other hand, if the first results are available after a short time and easy to read and interpret, you will find a lot of happy people around you that will support you for the rest of the project. This, plus the fact that you don't have to spend hours and hours to get ready for the next test run, is the real benefit of baseline tests. At the same time you also get the actual baseline results for the tests and can report on this immediately.

Preparation for baseline tests doesn't take as long as preparation for load tests. Test data is not needed in large volumes and the hardware and software requirements to execute the tests are easily achieved. The results are easy to read and reports can be done very quickly. If a problem shows up during the

baseline testing this is identified easily and can be isolated at the same time. Isolating problems that show on busy graphs usually takes a long time, but with baseline tests a lot of time is saved with this as well.

Executing Baseline Tests

Ideally baseline tests should be done with 1, 2, 5, 10, 20 and 50 users. This should be done for each script that will be part of the load test individually and immediately isolates any problems that might occur. The baseline tests should run for 20 to 30 minutes to get useful averages. Fight for time to do this and make it part of your script and load test preparation. The baseline tests give everyone a clear view of system performance and save a lot of time with identifying and isolating the causes of problems. Large volumes of test data is not lost with every failed test run and preparation for the next test can be completed quickly. Use these facts to motivate the need for proper baseline tests. When done properly, you'll need a lot less time for the load testing.

The goal to work for is to start the first load test only after all the problems identified during baseline testing have been resolved. This opens the opportunity to have an almost perfect run when executing the first load test. Errors encountered during the load test will nearly all be load related with other problems already fixed during the baseline tests.

Monitoring

As with all performance testing, proper monitoring should be done during the baseline testing. Without proper monitoring you often have to repeat a test if a problem is encountered to be able to identify what is wrong where. I always keep a close eye on monitoring throughout the performance testing. This gives me a very good feel for the system and I can pick up very quickly if something is not right at any point during any test. Include all the responsible people right from the start. Meaningful results can't be extracted without proper monitoring.

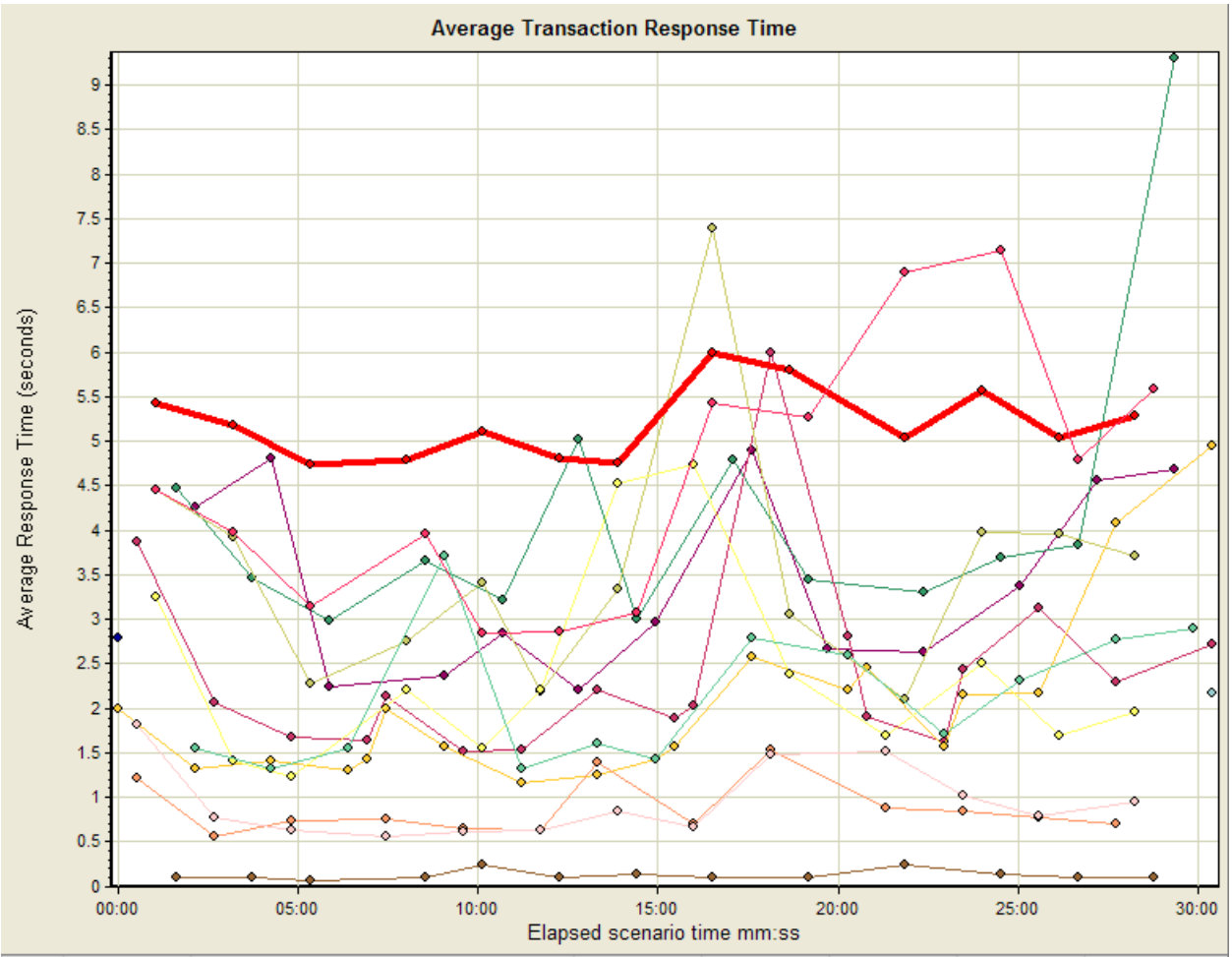
Examples – Baseline Test Results

Let's look at some examples to put everything into perspective.

The first series of graphs show average transaction response times for a script that captures customer details. The graphs are for 1, 5, 20 and 50 users. In this example a problem was identified with a specific step, Search Postal Code, which had to be fixed before the actual load test with 500 users could be done. The data preparation for this script was horrendous with a lot of unique data that could only be used once. Starting with a 500 user load test would have caused many headaches.

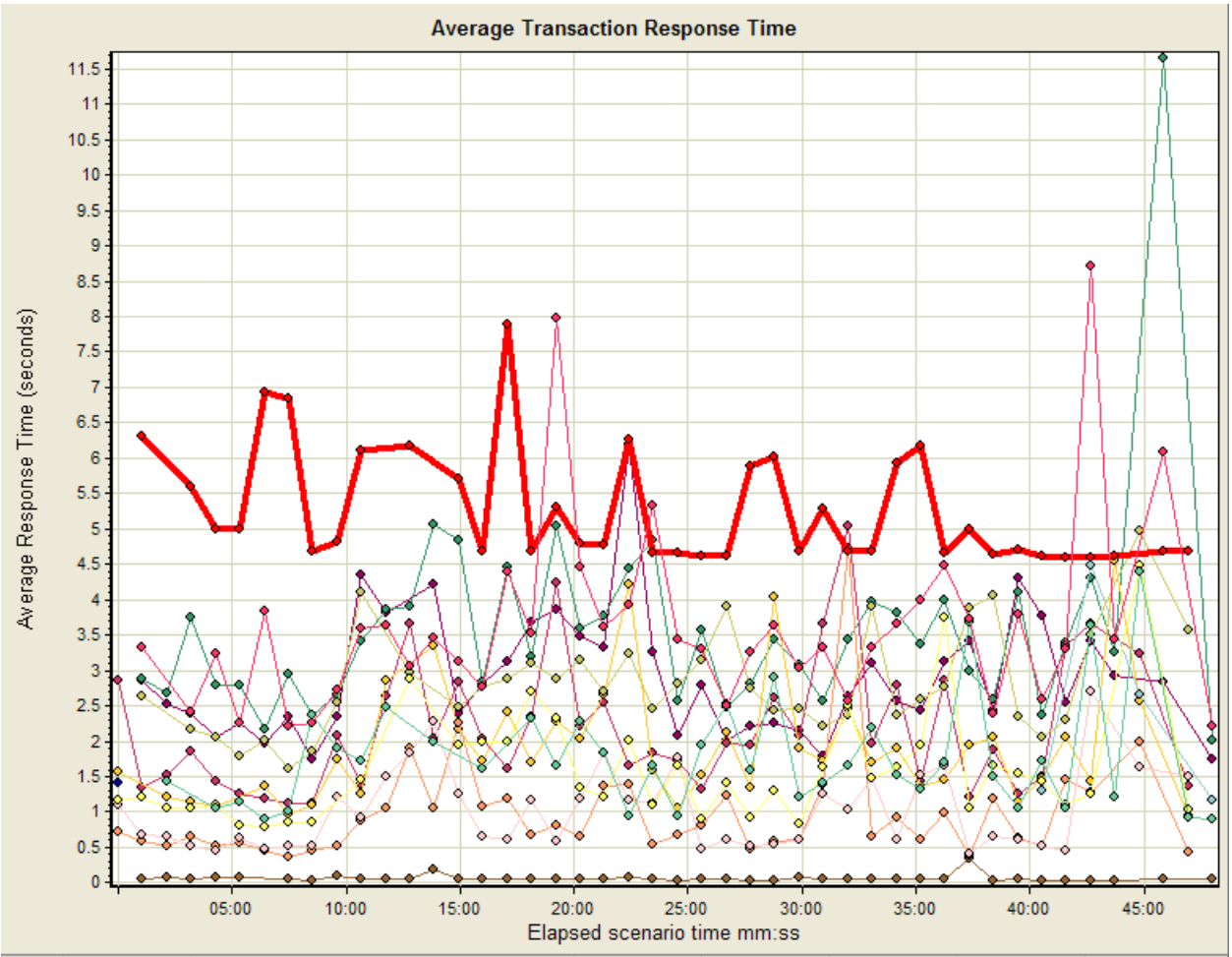
The Search Postal Code transaction is highlighted in red on each graph.

Capture Details – 1 User



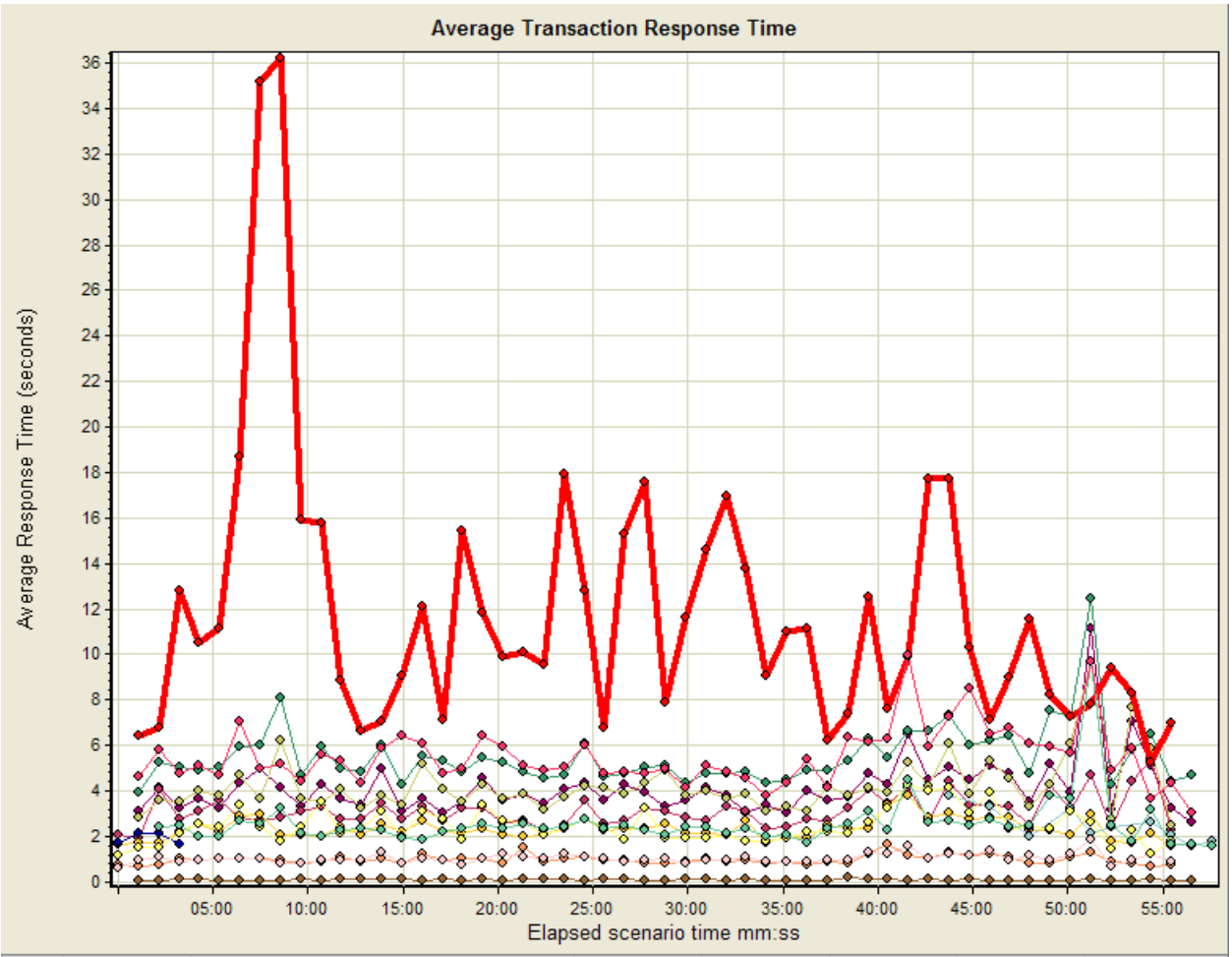
Color	Scale	Measurement	Graph's Minimum	Graph's Average	Graph's Maximun	Graph's Median	Graph's Std. Devia
	1	Client Affordability	2.213	3.42	4.894	2.961	1.012
	1	Client Contact Details	2.977	4.167	9.305	3.662	1.608
	1	Client Search	1.512	2.411	5.985	2.135	1.053
	1	EHL Menu	1.169	2.03	4.941	1.574	0.956
	1	Login	2.797	2.797	2.797	2.797	0
	1	Logoff	2.166	2.166	2.166	2.166	0
	1	Open Client Address Search	0.062	0.119	0.234	0.094	0.052
	1	Open Employ Search	0.561	0.872	1.527	0.748	0.293
	1	Registration New Loan	2.104	3.58	7.388	3.413	1.318
	1	Search Employer	0.561	0.944	1.821	0.795	0.391
	1	Search Postal Code	4.738	5.193	6.001	5.112	0.391
	1	Select Employer	1.231	2.411	4.738	2.213	1.076
	1	Submit Offer	1.325	2.12	3.709	1.714	0.739
	1	Who Is Employer	2.837	4.57	7.138	4.457	1.394

Capture Details – 5 Users



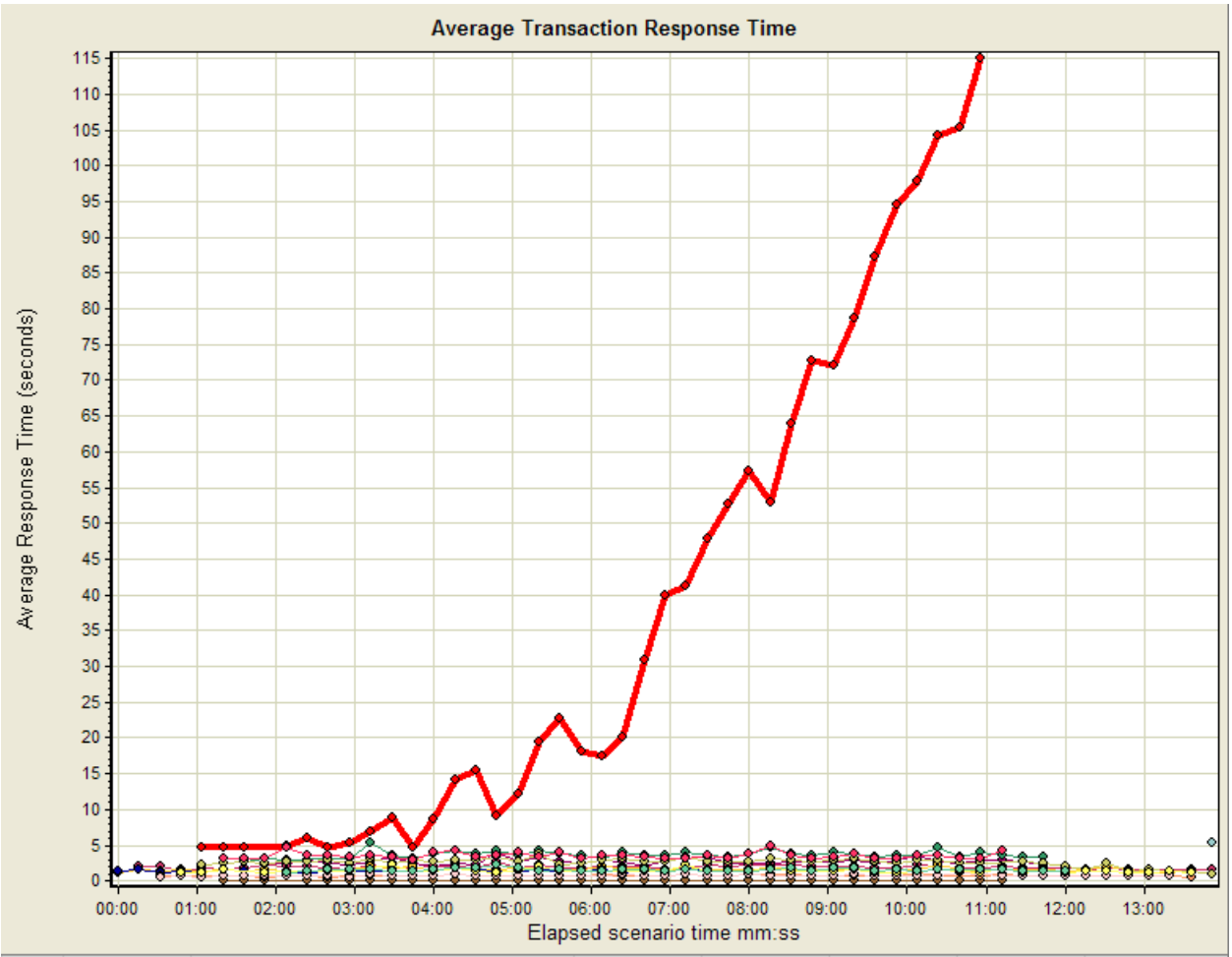
Color	Scale	Measurement	Graph's Minimum	Graph's Average	Graph's Maximun	Graph's Median	Graph's Std. Devia
	1	Client Affordability	1.749	2.895	6.208	2.79	0.873
	1	Client Contact Details	2.015	3.582	11.651	3.389	1.471
	1	Client Search	1.112	2.179	5.05	1.96	0.912
	1	EHL Menu	0.937	1.93	4.545	1.723	0.851
	1	Login	1.407	1.407	1.407	1.407	0
	1	Logoff	1.171	2.401	4.482	2.655	1.335
	1	Open Client Address Search	0.031	0.066	0.351	0.062	0.051
	1	Open Employ Search	0.376	0.974	4.725	0.687	0.728
	1	Registration New Loan	1.62	2.817	4.982	2.71	0.726
	1	Search Employer	0.406	1.012	2.694	0.687	0.55
	1	Search Postal Code	4.592	5.256	7.899	4.795	0.811
	1	Select Employer	0.784	1.585	4.482	1.417	0.784
	1	Submit Offer	0.89	1.778	4.404	1.645	0.803
	1	Who Is Employer	2.207	3.618	8.715	3.432	1.325

Capture Details – 20 Users



Color	Scale	Measurement	Graph's Minimum	Graph's Average	Graph's Maximun	Graph's Median	Graph's Std. Devia
	1	Client Affordability	2.6	4.147	11.151	3.867	1.286
	1	Client Contact Details	3.942	5.521	12.47	5.075	1.325
	1	Client Search	1.924	3.117	5.37	3.071	0.694
	1	EHL Menu	1.555	2.333	3.778	2.256	0.446
	1	Login	1.662	1.921	2.145	2.124	0.216
	1	Logoff	1.574	2.496	3.797	2.43	0.688
	1	Open Client Address Search	0.057	0.096	0.181	0.09	0.021
	1	Open Employ Search	0.656	0.978	1.649	0.975	0.194
	1	Registration New Loan	2.497	4.222	9.699	3.894	1.225
	1	Search Employer	0.611	1.034	1.891	0.981	0.207
	1	Search Postal Code	5.25	11.803	36.209	10.3	6
	1	Select Employer	1.185	2.453	4.252	2.344	0.717
	1	Submit Offer	1.563	2.562	9.716	2.375	1.121
	1	Who Is Employer	3.047	5.511	9.985	5.142	1.298

Capture Details – 50 Users



Color	Scale	Measurement	Graph's Minimum	Graph's Average	Graph's Maximun	Graph's Median	Graph's Std. Devia
	1	Client Affordability	2.082	2.72	3.997	2.613	0.402
	1	Client Contact Details	2.66	3.683	5.413	3.624	0.523
	1	Client Search	1.271	1.918	2.746	1.874	0.238
	1	EHL Menu	1.093	1.597	2.311	1.579	0.231
	1	Login	1.03	1.41	1.895	1.404	0.193
	1	Logoff	5.244	5.244	5.244	5.244	0
	1	Open Client Address Search	0.052	0.07	0.133	0.062	0.018
	1	Open Employ Search	0.5	0.723	1.129	0.692	0.118
	1	Registration New Loan	0.984	2.586	3.713	2.699	0.558
	1	Search Employer	0.567	0.724	1.07	0.702	0.103
	1	Search Postal Code	4.68	38.574	115.04	20.043	35.033
	1	Select Employer	1.046	1.417	2.356	1.328	0.272
	1	Submit Offer	1.248	1.561	2.317	1.529	0.202
	1	Who Is Employer	2.928	3.521	4.926	3.474	0.438

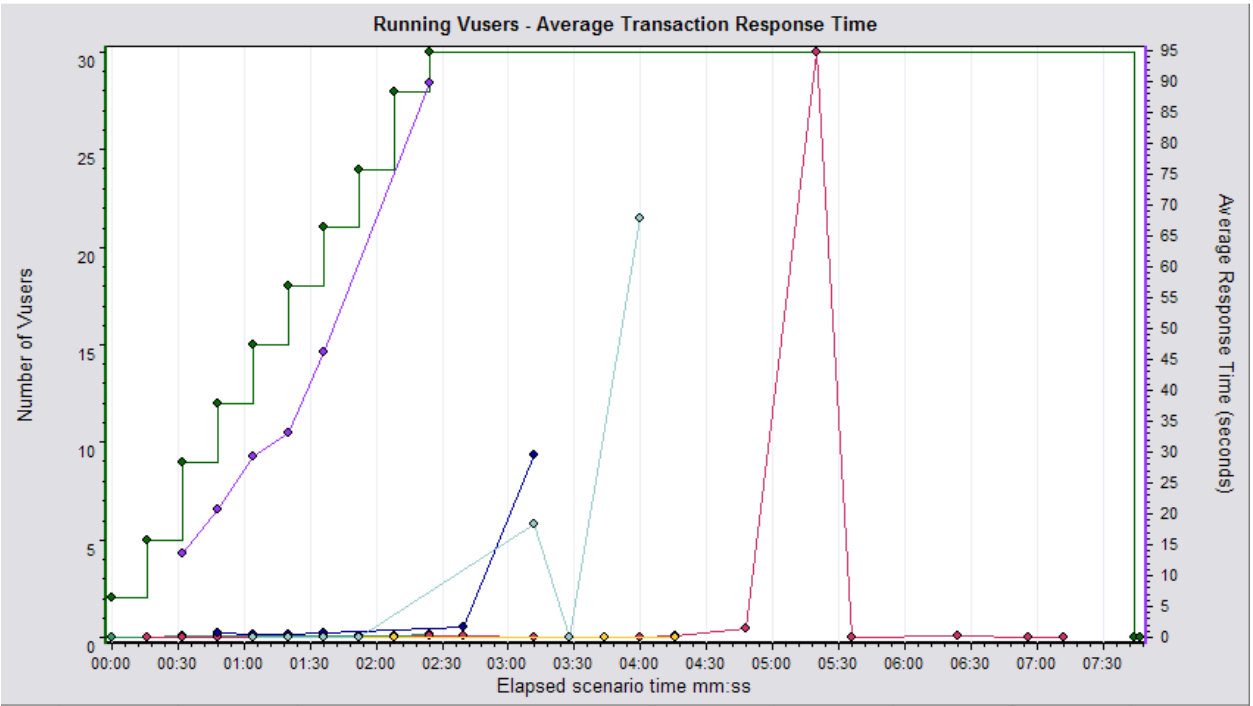
The degradation in performance from 1 to 50 users can clearly be seen. With 50 users the system eventually crashed. The advantage of the baseline tests are graphs with limited measurements where the problem transaction can be easily identified. If this was run with other scripts and numerous other measurements, isolating the problem could have been really difficult.

These results were available early in the project and justified the importance of the performance testing immediately. Looking good early on helps to gain respect and trust from the right people for the rest of the project.

The next example shows the importance of having monitoring in place during the baseline testing. In this example a search function caused the system to crash with 30 users. The requirement was to have a successful 1000 user test at the end of the performance testing. The time saved by running baseline tests for each individual script on this specific project was enormous. Problems as the one in the example were identified in 3 out of 7 scripts. If all the scripts were run together and with load, finding and isolating the problems responsible for the system crash would have been very difficult and time consuming.

The 30 user test for the search function failed when the users received time-outs as can be seen on the graph below. The graph shows the transaction response times with the running users.

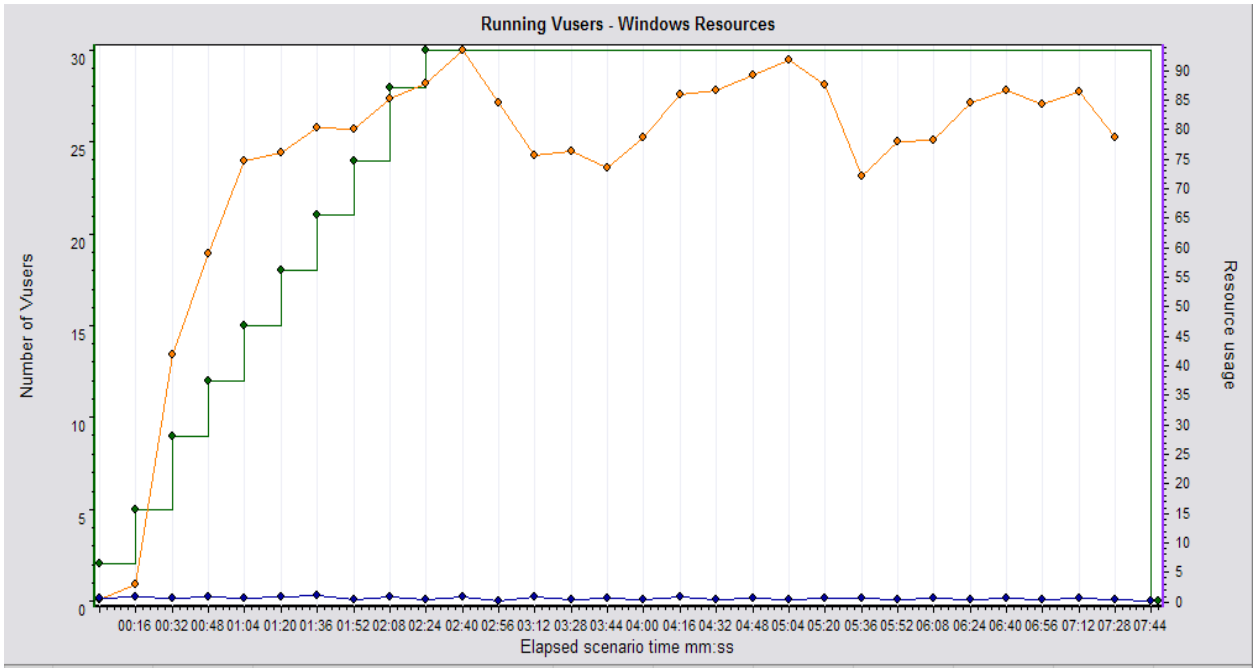
Running Users vs. Average Response Time (users left axis, response times right axis)



Color	Graph	Scale	Measurement	Graph's Minimum	Graph's Average	Graph's Maximun	Graph's Median	Graph's Std. Dev
	Running Vusers	1	Run	0	13.667	30	15	10.258
	Average Transac	1	Find	13.652	38.764	89.704	33.041	24.929
	Average Transac	1	Login	0.104	0.22	0.419	0.216	0.088
	Average Transac	1	Srch-Tran	0.041	4.921	94.701	0.132	20.599
	Average Transac	1	View Audit Trail	0.075	0.094	0.129	0.088	0.018
	Average Transac	1	View Primary Info	0.562	5.638	29.549	0.74	10.699
	Average Transac	1	View Transaction History	0.092	12.399	67.923	0.12	23.523

The system resources showed very high CPU usage on the application server when users performed the search function. The graph below shows the CPU utilization of the application server as well as the database server during the 30 user test that failed. Note the low CPU usage on the database compared to the application server

Running Users vs. System Resources (users left axis, CPU percentage right axis)



Color	Graph	Scale	Measurement	Graph's Minimum	Graph's Average	Graph's Maximun	Graph's Median	Graph's Std. Dev	Machine Name	Monitor Type
	Running Vusers	1	Run	0	13.667	30	15	10.258	N/A	N/A
	Windows Resou	1	% Processor Time (Processor _Total):10.182.198.1	0.438	74.492	93.439	80.09	22.148	10.182.198.116	Windows Resou
	Windows Resou	1	% Processor Time (Processor _Total):10.182.198.4	0.195	0.631	1.052	0.621	0.216	10.182.198.45	Windows Resou

Resource monitoring helped to identify the problem quickly. As in the first example, these results could be reported on early and meaningful results were distributed soon after the start of the testing effort. The importance of having monitoring in place at all times is highlighted once again.

Summary

With numerous successes as shown in the examples above, baseline testing has proven itself as an essential part of the performance testing process. The time saved by finding and isolating problems early on is enormous. Showing good, meaningful results early on a project has many advantages. It highlights the importance of performance testing and showing the improvements when problems are fixed make the rest of the project a very positive experience.

Include baseline tests in your planning and leave enough time for this. Do it properly, don't take any shortcuts. The benefits are great and when the actual load testing begins, the system will already perform well. You will also know the system well by then and will be able to see changes in trends or performance immediately. This makes the testing just that much better and you a better performance tester.