



Performance Test Results Analysis

Charlie Audritsh
IWST February 2005



Performance Test Results Analysis

- Analyzing the results is more important than it may seem
- If you did not design your test well enough, the results can be bewildering and meaningless



Functional Automated Testing vs. Performance Testing

- **Functional Automated Testing:**
 - Tool expertise is important to write effective tests
 - Can approach the application as a black box
- **Performance Testing:**
 - Effective performance tests can be quickly developed without knowing *everything* about the tools
 - *Must* have some understanding of the architecture and the environment and what load may do to it



A poorly thought out test can make the results bewildering

- Lets increase load until the app breaks!
- Oh, wait a minute, what exactly does "break" mean?



What is meant by "finding the breaking point" of an application under load?

Does "Broken" mean:

- Response is unacceptably slow?
- A system crash or outages caused by load?
- How about if the app is mostly working fine, but the level of load prevents the final calculations from being completed, in which case the fact that the rest of the app is working is meaningless?
- How about the point at which the app becomes completely useless to the user? Aka, "Locked up"?



The answer I got...

was what I suspected:

"The load breaking point can be interpreted in any of the cases that you have mentioned."

If you don't know exactly what you mean by "break", how will you know when you've done it?



To make your results meaningful...

- Put adequate thought into your test
- It's all about coming up with meaningful metrics that are repeatedly measurable



A test thought through

- Had a monthly volume target of completed transactions the app should allow the users to achieve
- Worked backwards from this and considered other things, like the real production performance of similar existing systems
- Arrived at a goal for the system to be able to complete 10 transactions per minute



Test details

- Start 10 instances of the script per minute
- But how do you know if the application can *complete* 10 per minute?
- Starting 10 per minute, and running for an hour, should have 600 successfully completed
- Anything less than that, the system could not do it



But how well *did* it do?

- Hard to say... Because we did not hit the target, at some point the system slowed down
- The number successfully completed by this test does not represent the number of transactions per minute the system can currently handle
- Want to know that, so we have an idea of how far we have yet to go



Expanding on this approach

- Work up to 10 transactions per minute
- I ran separate tests shooting for 2, 3, 4, 5, 6, 7, 8, and 9 per minute
- The test that completed the highest expected number of transactions successfully, represented what the system could currently handle
- The first time out it was 4



Advantages of this approach

- Showed clearly what the system could handle and how far we had to go
- Also made the first problems encountered above that level easier to see



Advantages of this approach

- Must fix the first error first, because anything after that could just be cascading errors
- Only shooting for the ultimate goal and missing leaves the issues found as a confusing mass, making it really hard to determine what went wrong
- The better information you can give development, the more likely they are to find and fix the real problem



Conclusion (finally!)

- Successfully analyzing performance test results lies in making them analyzable
- Think the test you're planning through well