ANZ Testing Board
SIGIST

Lessons Learnt Performance Testing

Who are we

- Martin Corke
  - Senior technical tester
- Jamie Bate
  - Senior .NET developer
  - ALM Consultant

What were our questions

- What type of application is it?
- How is the application being used?
- What should be tested?
- Who is involved in the testing?
- How should the results be presented?
- What should be done with the results?
- Common gotchas.

What type of application is it?

- System makeup
  - Web
  - Client / server
  - Desktop
- Do the users log in?
- Is the system stateless?
  - e.g. web / internet
- What is the environment makeup
  - LAN / WAN combination?

How is the application being used?

- How many concurrent users are there likely to be:

  Our testing: a concurrent user was a user currently logged in and actively performing transactions. Throughput may be defined as the number of completed work items per hour – the number of typical transactions can then be estimated and thereby giving an estimate of the pacing required and the sequencing of the transactions.

- What will be the frequency of use.

  Are there any peaks? Do you need to think about times when your system will be under higher loads such as timesheets or end of month billing.

- For existing applications this information will usually be readily available.

  Our testing: There is an existing application but only estimates were available along with interrogation of existing client database to determine times when certain record creations were highest.

What should be tested?

- Looking at
  - Excessive network chattiness
  - Database inefficiencies
  - Service layer, use of caching?
  - Business critical transactions, requires definition.
  - Establish concurrency, transaction throughput – peaks, average
  - Scalability requirements, requires definition.

- Obtain an initial benchmark.

  Our testing: Single user test to measure the transaction times for a single-user with no load on the system . This is the best case scenario and can be used very early on in the development – at the earliest point where there is some stability in the system. It may pick up some early performance issues such as inefficient SQL / stored procedures, excessive network chattiness, security issues e.g.. NTLM fall-back if unsuccessful authentication via Kerberos

How should it be tested?

- Test matrix:

  Important to develop a test matrix for business critical transactions. These can be used for benchmarking single-user tests (or n+1th tests – see above). Define start and stop events, location of user relative to the system (i.e. Include latency and bandwidth).

  These tests can be done manually and give a quick indication of possible reversal of performance per release.

- Stopwatch tests:

  Simple stopwatch tests are useful for gathering information for a "real" user experience but they must be done consistently and as mentioned earlier, the test environment prerequisites must be established and the start and stop events must be clearly defined

Part 2 -How should it be tested?

- Our testing:

  Estimates of concurrent users and average and peak throughput – stress testing was run initially to establish where the initial bottleneck occurred – i.e. Where the system breaks down. The number of users was ramped up slowly until transaction errors occurred.

  Then a series of load tests – ramp up to a defined number of concurrent virtual users and use the required transaction pacing to establish the required transaction throughput.

  In addition have a "real" user (the n+1th) user doing end-user transactions and recording the transaction time as the end-user would experience when the system is under load (transaction start and end events are defined and the user records the time between these events using e.g. stopwatch).

Who is involved in the testing?

- Cross functional team effort all typical with other responsibilities too.

- Business Analyst:
  - Define the business critical transactions.
  - Establish concurrency, transaction throughput – peaks, average
  - Scalability requirements
  - Review results

- Architect:
  - Define key performance indicators: network, servers
  - Scalability requirements
  - Analyse and review results

- Developer/Technical Tester:
  - Develop and maintain test scripts
  - Maintain test environment
  - Execute load tests
  - Analyse and review results

- IT Support:
  - Maintain test environment

- DBA:
  - Analyse and review results – database server

How should the results be presented?

- Who is the audience for these results?
  - Information for Service Level Agreements.
  - Internal audience.

- The results need to be understandable.
  - Are things getting better or worse?
  - How much?
  - Is 200% better actually mean anything?

- A summary page is worth it's weight in gold
  - e.g. Visual Studio, gives information at-a-glance and highlights slowest transactions, errors, average transaction times.

- Annotated Graphs/trend analysis, this is the easiest to convey information

What should be done with the results?

- Start with the errors – find out where the errors start occurring and what is the root cause of the errors. This analysis might require a refinement and re-run of the test to help isolate the errors and understand the nature of the error.

- Look for the “Knee” in transaction response times as the number of users (and transaction rates) increase – the transaction response time may stay fairly flat as the number of virtual users increases then become unstable and rise sharply, often accompanied with errors (the “Knee”).

- Given to the appropriate people to analyse and possibly act on.
  - Architects and/or the senior engineers

- Look at worst offenders and concentrate on those
  - Quick wins .... or is there a bigger underlying issue.

Common gotchas.

- Database
  - Poor indexing
  - Indexes not maintained
- Appropriate sizing of the environment – ideally something similar to production environment but not usually achievable.
- Highly customizable environments (database) – performance issues may go undetected because of user customizations.
- Appropriate database sizing and relevant (real-world) user data. For new applications this can be problematic. If the production environment includes WAN access then ensure that information on latency and bandwidth restriction is available and model this in the test scenarios.