

Performance Testing for Managers

Presented by Stuart Moncrieff
at SIGiST Melbourne on June 15th, 2011



What will be covered?

- Performance Testing as it applies to:
 - Large multi-user enterprise IT applications
 - Large websites
- Out of scope
 - Scalable architectures
 - Performance tuning tips
 - Types of test cases
 - Comparison of testing tools
- In Scope
 - Risks
 - Schedule
 - Resourcing



High-Profile Failures

CFA computer failures raise fresh alarm on day of danger

Andra Jackson
December 17, 2009

Comments 2

FOR EMERGENCIES DIAL '000' Home | Contact | Publications | News & Events | Sitemap | Links | Members

CFA

Fire & Incidents Fire Restrictions Joining CFA About CFA Residents Business & Industry Local Government Students & Kids Teachers

Warnings & Advice Information Hotline Community Meetings

Incident Summary

ERROR

The requested URL could not be retrieved

While trying to retrieve the URL: http://osom.cfa.vic.gov.au/public/osom/osom_wrapper.html

The following error was encountered:

- Connection to 125.7.99.179 Failed

The system returned:

(119) Connection timed out

The remote host or network may be down. Please try the request again.

'Error' ... The CFA incident summary web page at 2pm yesterday.

FEARS that Victoria remains highly vulnerable to another Saturday disaster have been raised after the Country Fire Authority's online bushfire warning system failed twice ye

My School site fails on first morning of operation

Miki Perkins and Will Brodie
January 28, 2010

Comments 70

THE AGE
theage.com.au

'Massive fail': technical glitches upset ultranet training day

Megan Levy
August 9, 2010

A training day for teachers to familiarise themselves with a new virtual-classroom project has been hit with major technical glitches, with many staff unable to log on to the new system.

Victorian state-school students were given the day off today so their teachers could be trained on ultranet - a \$77 million online network that will give parents round-the-clock access to their children's lessons, homework, results and attendance.

However, many of the state's 42,000 teachers and principals experienced problems when they attempted to log on this morning, while those who could access the system reported that it was running extremely slowly.

Education Minister Julia Gillard defends the new My School website.

Education Minister Julia Gillard defends the new My School website. Video feedback | Video settings

ral Government's controversial My School website is off ed start after experiencing technical difficulties on its ing of operation.

Do You Need to Test?

- What are the consequences if it doesn't work under load? What is the risk?
 - Lives lost?
 - Financial loss?
 - Reputation damage?
- Good reasons for not testing:
 - If it doesn't work, it doesn't matter
- Bad reasons for not testing:
 - Our vendor told us that it will be fine
 - We have bought a lot of servers



The #1 Performance Myth

Buying more hardware solves all performance problems!



- All load/performance defects due to one of...
 - **Code:** inefficient code, slow algorithms, bad SQL queries, deadlocking, thread-safety, memory leaks.
 - **Config:** application configuration settings, pool sizes (threads/worker processes, database connections), web server settings, database tuning.
 - **Capacity:** number of servers, server sizing (CPU, memory, etc).

Functional Testing	Performance Testing
Most defects are due to Code. Defects are mostly handled by development team.	Defects due to one of: Code, Capacity, or Configuration. Many teams responsible for fixes.
Defects can be reproduced by developers (if given steps).	Defects usually impossible to reproduce without special tools.
Requirements well specified (comparatively)	Requirements usually poorly defined or non-existent.
Easy to run many test cases in parallel. Hundreds/thousands of test cases.	Can run one test case at a time, with gap for analysis. Very small number of test cases.
Possible to fix defects in parallel with testing. Duration \approx test execution time + n.	Defects like an onion. One layer at a time. Duration = \sum defect resolution time + test execution time

- Questions:
 - Will adding more people make performance testing faster?
 - Is this a task for junior/inexperienced resources?
 - Does it make sense to locate testers remotely?

Problem: Doing it late



- Constraints
 - Must be done once some functionality is working.
 - A final run must be done after all other test phases have competed, to ensure that defect fixes have not broken performance.
 - Any code changes (due to performance fixes) will require functional regression testing.
- Difficult to predict how long the cycle will take (due to unpredictability of time required to fix defects).
 - This creates schedule risk. The later performance testing starts, the greater the risk of it getting on the critical path and pushing out the Go Live date.
 - Beginner mistake: Assuming that everything will work, and not allocating time in the schedule for fixing performance problems.

Problem: No budget for testing

- Allocate budget for
 - Performance test tools
 - Performance test environment
 - Performance testers
- Put it in perspective
 - What is the overall project budget?
 - What is the value of the project to the business at completion?
 - What is the cost of application failure?
- You maximise risk of schedule slip if you try to minimise the cost of performance testing
 - (do it at the end of the project only, do not allocate time for fixes, use the cheapest tool)



Problem: The wrong team



- Performance tester characteristics
 - Communication skills
 - Technical (not business) background
 - Must be able to write code
 - Experience in performance testing
- The wrong person
 - Someone from the dev team
 - A team supplied by the vendor
 - Grads
 - Someone who “knows another tool by the same company”.

Problem: Defect Ping Pong

- Load & Performance defects can take a long time to fix.
 - Responsible group not clear (developers, system admins, DBAs, network team, app server team, middleware team). This can lead to finger pointing
 - More complicated than functional defects. More difficult to determine root-cause.
- Tips
 - Fixing performance defects should have the same priority as fixing functional defects. Don't ignore performance defects "until later".
 - Make sure that someone from all the different teams has been engaged by the project to assist with defects.



Problem: No Workload Model

- Common (and wrong) statements:
 - “Just do 100 users worth of load”
 - “Just tell me the maximum load the site can handle.”

Peak Hour volumes

Business Process	Volume
Register	10
Search for product	10,000
Browse catalogue	2,000
Update details	100
Place order	3,600

- Where to get this information:
 - Capacity planning data
 - Logs from existing system
 - Interviews with business experts

Problem: Unrealistic Test Environment

- The more differences there are between the Performance Test environment and the Production environment, the less predictive power the tests will have.
- Don't bully inexperienced performance testers into extrapolating test results from a very small environment to a very large one.
- Don't assume scalability based on adding more hardware (remember Code and Config, not just Capacity).



Problem: Poor change management

- What did we test against?
- What tuning changes were made during testing?
- Did all the changes make it into Production?



Problem: Bad Contract with Vendor

- Milestone payments not dependent on resolution of performance defects.
- Warranty period expires before application sees Peak Load (e.g. staged rollout, or “Cup Day” type yearly event)
- Not specifying performance requirements in original NFRs. Oops, that's a change request.



Problem: Ignoring the final report

- Performance test results are one of many inputs that The Business uses to make a Go-Live decision.
- Maybe there is a valid reason to deploy a system that will not work under load.



Key Takeaways

- Before:
 - Get the contract right
 - Allocate a budget
 - Requirements: the project must define a workload model!
 - Build a realistic test environment
- During
 - Use the right team, start early.
 - Your plan should not rely on everything working perfectly the first time.
 - Allow time to fix defects, make someone responsible for fixing them. Don't leave them until later. Hope for the best, plan for the worst.
- After
 - Don't ignore the report
 - Adding more hardware usually won't save you (Code/Capacity/Config)
 - Remember to apply your changes to Production



Questions?



- My writing
 - <http://www.myloadtest.com>
 - <http://www.jds.net.au/tech-tips>
- Social media
 - <http://au.linkedin.com/in/stuartmoncrieff>
 - <http://twitter.com/StuartMoncrieff>