SQA DAYS#23

Software quality assurance days

International Conference of
Software Quality Assurance

sqadays.com

Alexander Podelko

Oracle. Stamford, CT, USA

Continuous Performance Testing
Myths and Realities

Minsk. May 25–26, 2018

"Traditional" Approach

- Load / Performance Testing is:
  - Last moment before deployment
  - Last step in the waterfall process
  - Large corporations
  - Expensive tools requiring special skills
  - Protocol level record-and-playback
  - Lab environment
  - Scale-down environment
  - Checking against given requirements / SLAs
  - Throwing it back over the wall if reqs are not met
  - …

Agenda

- ***Agile Development & Performance Testing***
- Continuous Performance Testing
- Performance Engineering Puzzle: Changing Dynamics

Agile Development

- Agile development should be rather a trivial case for performance testing
  - You have a working system each iteration to test early by definition.
  - You need performance engineer for the whole project
    - Savings come from detecting problems early
- You need to adjust requirements for implemented functionality
  - Additional functionality will impact performance

The Main Issue on the Agile Side

- It doesn't [always] work this way in practice
- That is why you have "Hardening Iterations", "Technical Debt" and similar notions
- Same old problem: functionality gets priority over performance

The Main Issue on the Testing Side

- Performance Engineering teams don't scale well
  - Even assuming that they are competent and effective
- Increased volume exposes the problem
  - Early testing
  - Each iteration
- Remedies: automation, making performance everyone's job

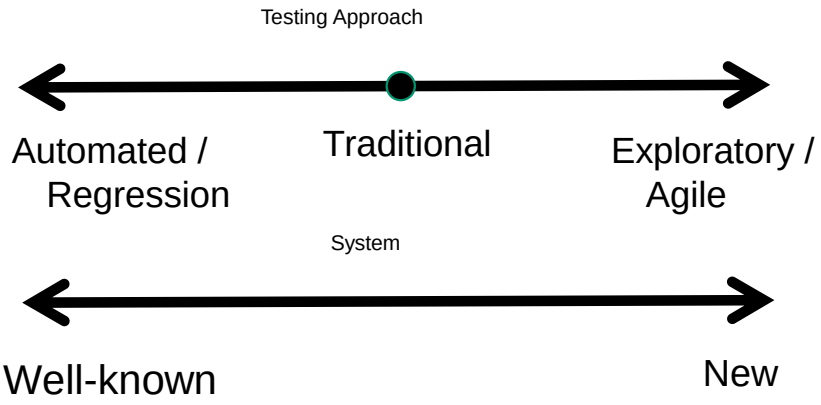
Mentality Change

- Making performance everyone's job
- Late record/playback performance testing -> Early Performance Engineering
- System-level requirements -> Component-level requirements
- Record/playback approach -> Programming to generate load/create stubs
- "Black Box" -> "Grey Box"

Exploratory Testing

- Rather alien for performance testing, but probably more relevant than for functional testing
- We learn about system's performance as we start to run test
  - Only guesses for new systems
- Rather a performance engineering process bringing the system to the proper state than just testing

Test Approach Dimension



Agenda

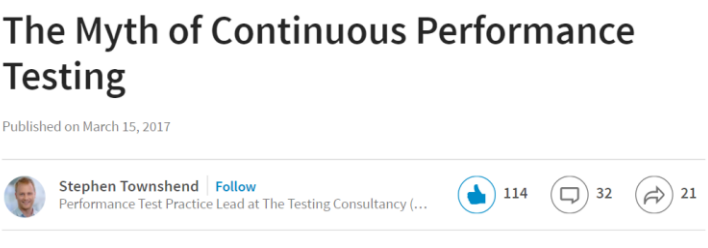
- Agile Development & Performance Testing
- ***Continuous Performance Testing***
- Performance Engineering Puzzle: Changing Dynamics

Myth or Reality ?

- You see Performance CI presentations at every conference nowadays….

and

- Still not many performance professionals do it
  - As far as I know….



Different Perspectives

- Consultant: need to test the system
  - In its current state
  - Why bother about automation?
  - External or internal
- Performance Engineer
  - On an agile team
  - Need to test it each build/iteration/sprint/etc.
- Automation Engineer / SDET / etc.

Automation: Considerations

- You need know system well enough to make meaningful automation
- If system is new, overheads are too high
  - So almost no automation in traditional environments
- If the same system is tested again and again
  - It makes sense to invest in setting up automation
- Automated interfaces should be stable enough
  - APIs are usually more stable on early stages

Automation: Difficulties

- Complicated setups
- Many parts of the puzzle
  - Long list of possible issues
- Complex results (no pass/fail)
  - Not easy to compare two result sets
- Changing/Fragile Interfaces
- Time / Resources considerations
  - Tests may be long / use a lot of resources

Complicated Setups

- [Assumption: we have basic elements in place]
- More complicated setups
  - Multi-machine
  - Keeping configuration [comparing apples-to-apples]
  - Larger/realistic set of data
  - Realistic security
  - Monitoring/instrumentation/logging setup

Many Parts of the Puzzle

- System Under Test
  - Usually distributed
- Load Testing Tool / Harness
- CI plumbing
- Results analysis / alerting
- And everything may go wrong
  - Needs extensive error handling
    - Which is a challenge between different tiers / tools

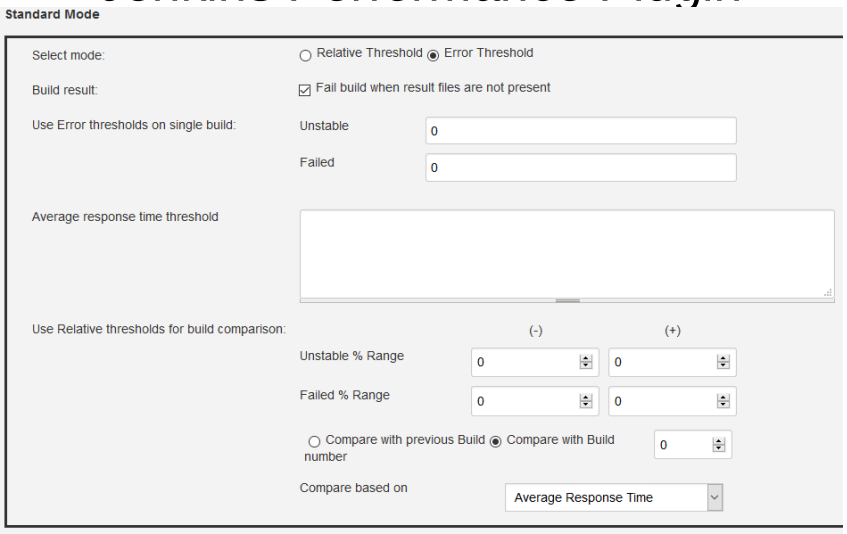
Continuous Integration: Tools

- CI support becoming the main theme
- Integration with Continuous Integration Servers
  - Jenkins, Hudson, etc.
  - Several tools announced integration recently
  - Making a part of automatic build process
- Automation support
- Cloud support
- Support of newest technologies

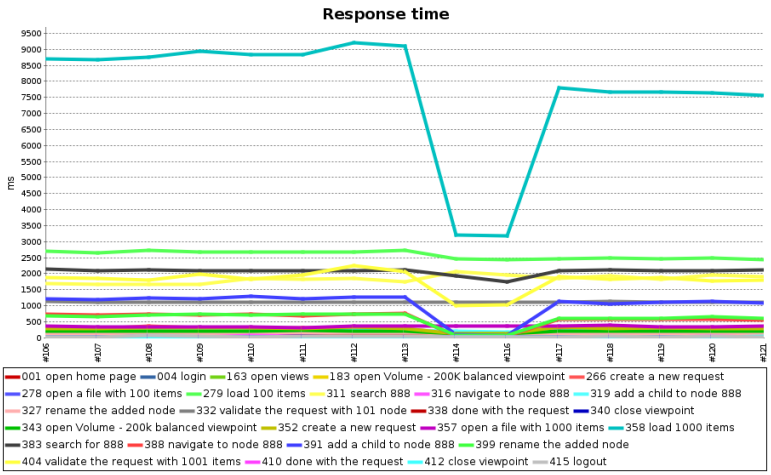
Complex Results

- No easy pass/fail
  - Individual responses, monitoring results, errors, etc.
- No easy comparison
  - SLA
  - Between builds
- Variability

Jenkins Performance Plugin

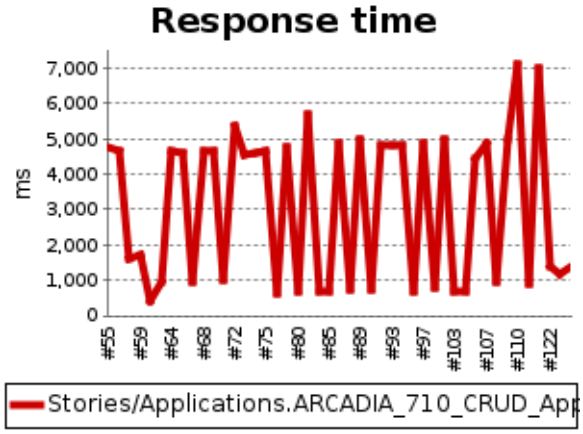


Performance Plugin



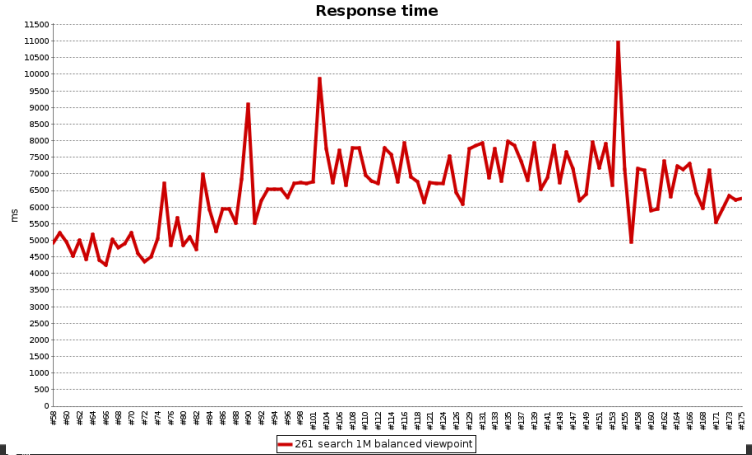
Variability - Environment

- Due to difference in environments



Variability - System

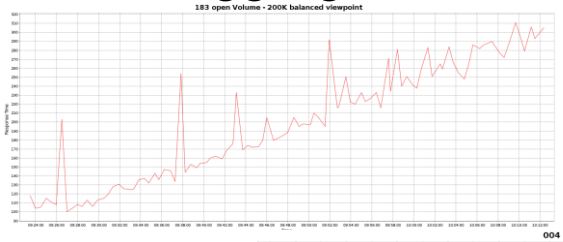
- Inherent to the test setup

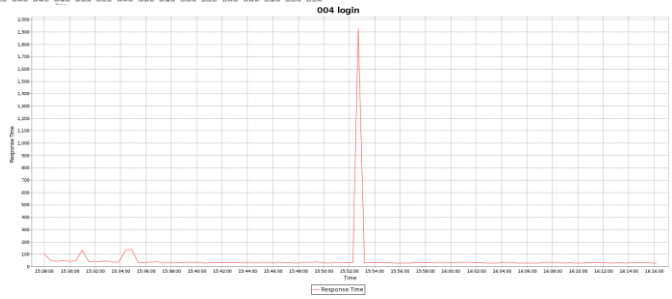


Variability - System

URI	Samples	Samples diff	Average (ms)	Average diff (ms)
001 home	1	0	347	-22
005 login	1	0	2438	-66
157 views	1	0	117	-33
173 open volume view	1	0	84792	3945
261 search 1M balanced viewpoint	1	0	10964	4295
262 navigate 1M balanced viewpoint	1	0	208	-47
268 open 1M flat viewpoint	1	0	17462	-1562
272 open 1M grid	1	0	5040	572
282 search 1M grid	1	0	2247	8
283 navigate 1M grid	1	0	8343	-181
286 open 200k balanced viewpoint	1	0	16890	-3703
289 search 200k balanced viewpoint	1	0	1261	-1027
290 navigate 200k balanced viewpoint	1	0	148	10
296 validate 200k viewpont	1	0	81126	723

Aggregation Challenge





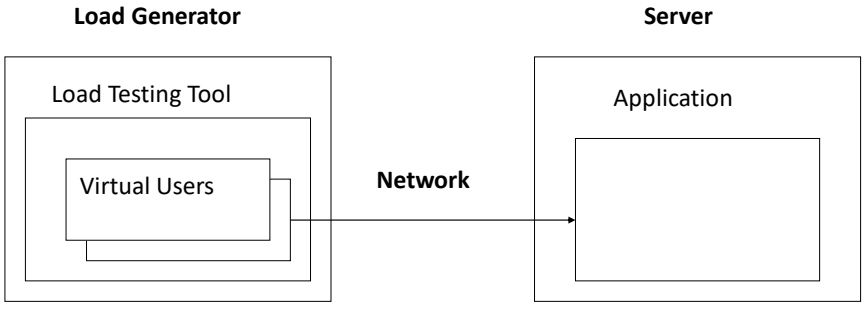
Addressing Variability

- Same environment / starting config
- No other load
- Multiple iterations
- Reproducible multi-user tests
  - Concurrent test (sync points)

Changing Interfaces

- If use recording, minor changes may break scripts
  - And you may even don't know that
  - Interface should be mature enough
- Not just protocol-level recording
  - GUI
  - API / Programming

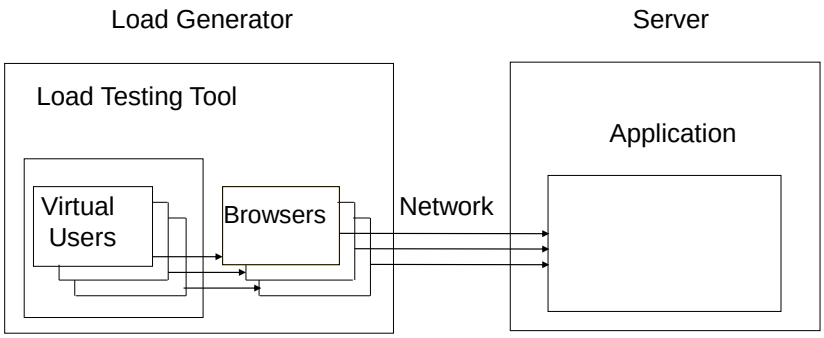
Record and Playback: Protocol Level



Considerations

- Usually doesn't work for testing components
- Each tool support a limited number of technologies (protocols)
- Some technologies are very time-consuming
- Workload validity in case of sophisticated logic on the client side is not guaranteed

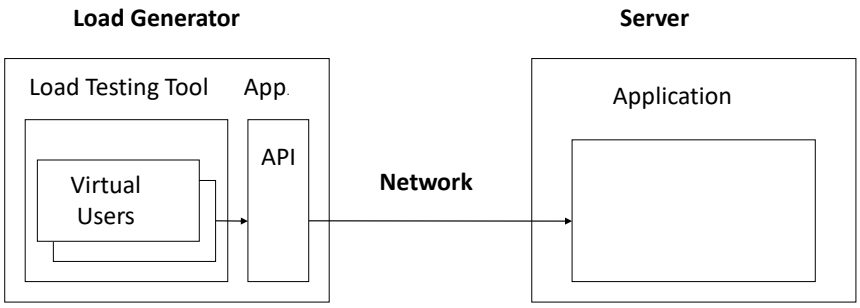
Record and Playback: UI Level



Considerations

- Scalability
  - Still require more resources
- Supported technologies
- Timing accuracy
- Playback accuracy
  - For example, for HtmlUnit

Programming



Considerations

- Requires programming / access to APIs
- Tool support
  - Extensibility
  - Language support
- May require more resources
- Environment may need to be set

Time / Resource Considerations

- Performance tests take time and resources
  - The larger tests, the more
- May be not an option on each check-in
- Need of a tiered solution
  - Some performance measurements each build
  - Daily mid-size performance tests
  - Periodic large-scale / uptime tests outside CI

Automation: Limitations

- Works great to find regressions and check against requirements
- Doesn't cover:
  - Exploratory tests
  - Large scale / scope / duration / volume
- "Full Automation" is not a real option, should be a combination

Myth or Reality?

- In the middle
  - Depends on context
- Reality in some cases
  - Usually stable systems / simpler test cases
  - Often single-user
  - Strong CI culture / CI expertise in house
- Still rather myth generically
  - Not much tool support for generic use

Agenda

- Agile Development & Performance Testing
- Continuous Performance Testing
- ***Performance Engineering Puzzle: Changing Dynamics***

Performance Risk Mitigation

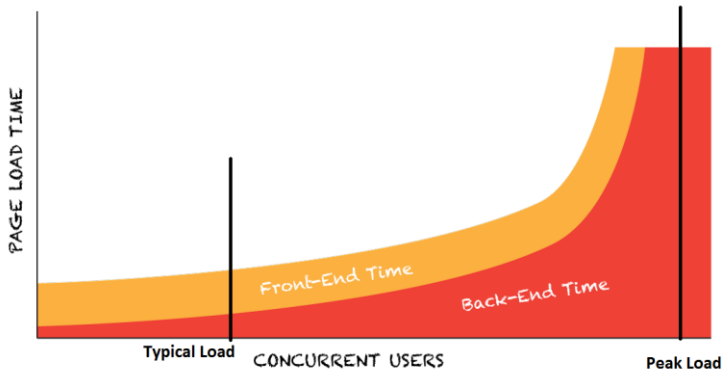
- Single-user performance engineering
  - Profiling, WPO, single-user performance
- Software Performance Engineering
  - Modeling, Performance Patterns
- Instrumentation / APM / Monitoring
  - Production system insights
- Capacity Planning/Management
  - Resources Allocation
- Continuous Integration / Deployment
  - Ability to deploy and remove changes quickly

But all of them don't replace load testing:

Load testing complements them in several important ways !

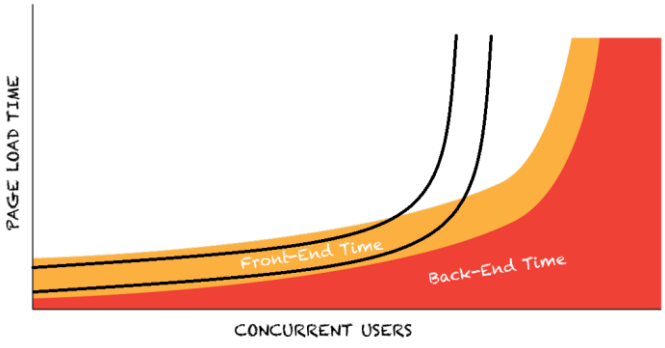
Can System Handle Peak Load?

- You can't know without testing:



Verify Multi-User Performance

- Single-user improvement may lead to multi-user performance degradation



What Else Load Testing Adds

- Performance optimization
  - Apply exactly the same load
  - See if the change makes a difference
- Debugging/verification of multi-user issues
- Testing self-regulation functionality
  - Such as auto-scaling or changing the level of service depending on load

Changing Dynamic / Historical View

- Mainframes
  - Instrumentation, Scheduling, Capacity Planning
- Distributed Systems
  - Load Testing, System Monitoring
- Web / Cloud
  - App Monitoring, Perf Engineering

So What Is Going On?

- I believe that load testing is here to stay, but should fully embrace the change
  - Not one-time, to become dynamic
- Dynamic of different PE approaches is changing
  - As it was during the whole history of PE
- Probably there would be less need for “load testers" limited only to running tests, but more need for performance experts who can see the whole picture using all available tools and techniques.

Summary

- CI becomes the main trend impacting performance testing
- It is a reality in simple cases
  - Some out-of-box tool support
- It is a lot of custom work in more complex cases
- Just a part of performance testing strategy
  - Important in iterative development

Questions?

Alexander Podelko

alex.podelko@oracle.com
alexanderpodelko.com/blog
@apodelko