Performance Testing in Agile Contexts

Increased productivity
Reduced costs
• @ericproegler, contextdrivenperformancetesting.com
• Performance Engineer SOASTA
• 13 years in (back-end) performance, 19 in software
• Speak Easy Mentor (speakeas.ie)
• AST Board (associationforsoftwaretesting.org)
• STPCon Advisory Board (stpcon.com)
• WOPR Organizer (performance-workshop.org).

SOASTA Ask

• Native app source code for being able to demo our mobile automation/RUM apps?

• You get Native RUM in prod.

• Talk later?
• No Process-teering/Scrubbaggery here.

• How can we make performance and scalability feedback faster...

• ...or at all, with incomplete/unfinished systems, on non-production hardware?

Performance Testing Iterations?
reliability
availability
serviceability
• **What are the risks we are testing for?**

• What is “Realistic”?  

• **Testing Techniques in Iterative Projects**

• **Performance Testing Incomplete Systems**
Scalability

Costly operations -> systems won’t scale well

• Wirth’s Law: “Software is getting slower more rapidly than hardware gets faster”

• Subject to use patterns and user models

What Does a Problem Look Like?

• Longer response times is a clue

• “High” CPU/Memory/Storage/Network Utilization

Performance Risks: Scalability
Capacity

System can’t structurally support expected load – baked-in serialization issues

What Does a Problem Look Like?

• Response time very sensitive to load
• Hard or Soft Resource Limitations
  • CPU/Network Limitation
  • Increasing I/O Latency
  • Database threads and other queues

Performance Risks: Capacity
Concurrency

Operations contend and collide (Race conditions, database locks, etc)

What Does a Problem Look Like?

• Infrequent functional issues that seem to only occur under load
• Process crashes
• Not easily reproducible (Heisenbugs?)

Performance Risks: Concurrency
Reliability

Degradation over time: system becomes slower, less predictable, or eventually fails.

What Does a Problem Look Like?

• Memory or Object Leaks
• More frequent Garbage Collection
• Decaying response times

Performance Risks: Reliability
• What are the risks we are testing for?

• What is “Realistic”?

• Testing Techniques in Iterative Projects

• Performance Testing Incomplete Systems
Will the **completed, deployed system support:**

(a, b...) users
performing (e, f...) activities
at (j, k...) rates
against mn... configuration
under rs... external conditions,
meeting x, y... response time goals?

(Simulation Test)

**Illusion of Realism: Experiment**
“All Models are wrong. Some are useful.”

- Guesses at activities, and frequencies
- Organic loads and arrival rates – limitations imposed by load testing tools
- Session abandonment, other human behaviors
- Simulating every activity in the system
- Data densities (row counts, cardinality)
- Warmed caching
- Loads evolving over time

Illusion of Realism: Models
“The environment is identical.”

• Shared resources: Virtualization/Public Cloud, SANs, Databases, Networks, Authentication...
• Execution environment versions and patching
• Software and hardware component changes, versions and patching
• Variable last-mile network conditions (mobile)
• Background processing, other activities against overlapping systems and resources

Illusion of Realism: Environment
instant access to reliable data
• What are the risks we are testing for?

• What is “Realistic”?

• **Testing Techniques in Iterative Projects**

• **Performance Testing Incomplete Systems**
Simulation Testing occurs at end of the project, just before go live. If we find anything, delay the whole project...

...or “Defer to Phase 2”

Simulation Tests are Expensive to Create, Maintain, and Analyze
How might “Realistic” translate to agile?

• Horizontal Scalability assumptions – let’s use them
• Test subsets: single servers, single components, cheaply and repeatedly
• Calibration tests

Cheap and Fast Performance Tests
Why must we have a “comparable” environment?

- Leverage horizontal scalability – one server is enough
- Who says we have to have systems distributed the same way? Why can’t my test database be on this system for calibration purposes?
- Isolation is more important than “Real”

Cheap and Fast Performance Tests
Repeatable, reliable, rapid tests:

- Login (measure session overhead/footprint)
- Simple workflows, avoid data caching effects
- Control variables
- Be ready to repeat/troubleshoot when you find anomalies

Cheap and Fast Performance Tests
Who needs “Real”? Let’s find problems.

Burst loads – Ready, Set, GO!!!

- No ramp, no think time, no pacing
- 10 threads, 10 iterations each
- 1 thread, 100 iterations
- 1 or 10 threads, running for hours/days

Cheap and Fast Performance Tests
Calibrate, and Recalibrate as necessary

- Environments
- Builds
- Day/times

...any time you want to be sure you are comparing two variables accurately

Calibrate Instruments
<table>
<thead>
<tr>
<th>Date</th>
<th>Build</th>
<th>Min</th>
<th>Mean</th>
<th>Max</th>
<th>90%</th>
</tr>
</thead>
<tbody>
<tr>
<td>2/1</td>
<td>9.3.0.29</td>
<td>14.1</td>
<td>14.37</td>
<td>14.7</td>
<td>14.6</td>
</tr>
<tr>
<td>3/1</td>
<td>9.3.0.47</td>
<td>37.03</td>
<td>38.46</td>
<td>39.56</td>
<td>39.18</td>
</tr>
<tr>
<td>8/2</td>
<td>9.5.0.34</td>
<td>16.02</td>
<td>16.61</td>
<td>17</td>
<td>16.83</td>
</tr>
<tr>
<td>9/9</td>
<td>10.0.0.15</td>
<td>17.02</td>
<td>17.81</td>
<td>18.08</td>
<td>18.02</td>
</tr>
<tr>
<td>10/12</td>
<td>10.1.0.3</td>
<td>16.86</td>
<td>17.59</td>
<td>18.03</td>
<td>18</td>
</tr>
<tr>
<td>11/30</td>
<td>10.1.0.38</td>
<td>18.05</td>
<td>18.57</td>
<td>18.89</td>
<td>18.81</td>
</tr>
<tr>
<td>1/4</td>
<td>10.1.0.67</td>
<td>18.87</td>
<td>19.28</td>
<td>19.48</td>
<td>19.46</td>
</tr>
<tr>
<td>2/2</td>
<td>10.1.0.82</td>
<td>18.35</td>
<td>18.96</td>
<td>19.31</td>
<td>19.24</td>
</tr>
</tbody>
</table>

**Calibration Results for Web Login, Search, View.**

*Burst 10 threads iterating 10 times*
Now the cost of a test is low, let’s run many

- Run five tests and average the results
- Run tests every day
- Run tests in Continuous Integration?

Cheap and Fast Performance Tests
• Who’s perf testing in CI?

• Wanna tell us about it?
• Stopwatch?
• Screen Captures? Videos?
• Waterfall Charts? Fiddler?
• Use Background Load and take measurements?
• Use/extend automation – Add timers, log key transactions, build response time records

• Watch for trends, be ready to drill down. Automation extends your senses, but doesn’t replace them

Extending Automation
• Supplement Load Tests with other automation.

• Load + Automation = ..... 

• Capture Waterfalls

Extending Automation
programs and packages
• What are the risks we are testing for?

• What is “Realistic”? 

• Testing Techniques in Iterative Projects

• Performance Testing Incomplete Systems
Test what is there:

- Session models: login/logoff are expensive, and matter for everyone
- Search Functions
- Think about user activities like road systems – where are the highways?
- What will be meaningful to calibrate?
• APIs – abstraction points, less brittle than interface

• Mocking/Stubs/Auto-Responders/Service Virtualization – built in abstraction points to simulate missing/in-progress components
Incomplete Systems:

Business Logic
Incomplete Systems: Business Logic

• Frequently the scaling bottleneck in this architecture pattern. Web services are often big buckets of functionality, in interpreted code

• Abstract and borrow code from presentation layer, test the rest of the system down

• Script load tests that address web services directly
Incomplete Systems:
Authentication
Create Sessions

• Response time to create a session?
• How reliably can I create sessions?
• How many sessions can I create?
• What arrival rate can I support?
Find or Make a Test Harness.

• Response time to push a message?
• Response time to pull a message?
• Is this sensitive to load?
• At what rate can I create messages? With one publisher? With multiple publishers?
• At what rate can I consume messages? What if I add message servers/dispatchers?
improved...
data storage capability
data access
superior technology
• What are the risks you are testing for?
• Unpack “Realistic”: what does it mean, and what (who?) is it for?
• Make Performance tests that are cheap and repeatable
• Calibrate!
• Test What You Have

Some Conclusions?
Please Evaluate?

@ericproegler
performance-workshop.org

See you next time!