Performance Testing: Throughout the Application Life-Cycle

Created for:

By:
Scott Barber
Chief Technologist
PerfTestPlus, Inc.
Performance Testing: Throughout the Application Life-Cycle

Scott Barber

Chief Technologist, PerfTestPlus, Inc.
sbarber@perftestplus.com
www.perftestplus.com
@sbarber

Co-Founder: Workshop On Performance and Reliability
www.performance-workshop.org

Author: 
Co-Author: 
Contributing Author:

Books: www.perftestplus.com/pubs
About me: about.me/scott.barber

© 2012 PerfTestPlus, Inc. All rights reserved.
This keynote is a sub-set of 7 days of instruction (about the same as a 1 semester university course)

I do not believe in “Best Practices” (I believe in problem solving via experience, experiment & education)

Everything in this keynote is based on personal experience in *some* context (but not yours)

Adapt concepts to your situation/context

Do network with others who are here (I bet they have great ideas too)

Follow–up with me (what works for you… or doesn’t)

I *like* questions!
Background
What is Performance?

System or application characteristics related to:

Speed:
- responsiveness
- user experience

Scalability:
- capacity
- load
- volume

Stability
- consistency
- reliability
- stress
What is Performance Testing?

What mom tells people:
I help people make websites go fast.

What I tell people:
I help and/or teach individuals and organizations to optimize software systems by balancing:

• Cost
• Time to market
• Capacity

while remaining focused on the quality of service to system users.
Performance vs. Load Testing?

Performance is to Load as Rectangle is to Square
Who is Responsible?

Everyone
Where Does it Fit?

Everywhere
Allow me to explain...
To Prevent Poor Performance...

To Prevent a Wreck

Left and Right
You Must Check

... don’t just react when it happens.
An Ounce of Prevention...

“What fits your busy schedule better, exercising one hour a day or being dead 24 hours a day?”
Which implies that...
The Performance Lifecycle is: **Conception to Headstone**

Not **Cradle to Grave**
But what does that **mean** ?!?
... Do you like Swim Lanes?

Performance Strategy Goal Concept; Program Level; Release n+1 thru n+2

Executives
- Oversee, Review and/or Approve

Developer(s)
- Collect Stories for n+2
- Add Performance & Profiling Stories
- Develop Stories; Profile; Perf Enhanced Profiling
- Troubleshoot / ID Tuning Opportunities

Architect(s)
- Design n+2
- Architect & Model n+2
- Continuous Iteration

ITSD/Ops
- Build Capacity Model for n+2
- Baseline n+1 in Prod
- Reverse Cross-Validate Release Readiness Perf Tests in Staging

Performance Test(s)
- Update Project Performance Strategy; Design/Develop Investigative Performance Tests
- Observe, Consult and Assist Where Appropriate

Sprints 1-X

Deploy

Provide Current Production Info & Models, Consult When Needed

Cross-Validate n+2 Capacity Predictions, Architecture Model Predictions, Perf Test Results in Staging

Validate n+2 Capacity Predictions in Prod

Observe, Advise, Troubleshoot, Validate and/or Update Architecture Model

Observe, Report Unacceptable Performance

System Investigative Performance Testing

System Readiness Perf Testing (System)
...Or maybe Circles?

Agile Performance Testing Activities
1. Understand Project Vision and Context
2. Identify Reasons for Testing Performance
3. Identify Value of Testing Performance
4. Configure or Update Tools and Load Generation Environment
5. Identify and Coordinate Tasks
6. Execute Task(s)
7. Analyze Results and Report
8. Revisit 1-3, Consider Performance Criteria
9. Reprioritize Tasks
...Relativistic Comparisons?

Distribution of Responsibility for System Performance

- Application Architect(s)
- Capacity Planner(s)
- Performance Tester(s)
- Developer(s)
- Operations Engineer(s)

Proportion of Total Performance Focus

Concept → R&D → Prototype → Productize → Deploy → Support →

Conceptual Phases
...How about Colors?

<table>
<thead>
<tr>
<th>Method</th>
<th>Design</th>
<th>Develop</th>
<th>Test</th>
<th>Deploy</th>
<th>Maintain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Waterfall</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Iterative</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Agile</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Continuous Integration</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Continuous Delivery</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DevOps</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Actual</td>
<td></td>
<td>Deploy</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ideal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Waterfall: Design, Develop, Performance, Test, Deploy, Maintain
- Iterative: Design, Develop, Test, Design, Develop, Test, Deploy, Maintain
- Agile: Sprint, Performance, Test, Performance, Test, Deploy, Maintain
- Continuous Integration: Sprint, Sprint, Sprint, Sprint, Performance, Deploy, Maintain
- Continuous Delivery: Sprint, Sprint, Sprint, Sprint, Sprint, Sprint, Sprint, Sprint, Performance, Sprint
- DevOps: Sprint, Sprint, Sprint, Sprint, Sprint, Sprint, Sprint, Sprint, Sprint
- Actual: Sprint, Deploy, Performance Panic, Sprint
- Ideal: Sprint, Sprint, Sprint, Vacation, Sprint, Optimized & Monitor, Sprint, Optimized & Monitor
I’ve tried all of those...
... But now I prefer...
...the Performance Puzzle.
Application Delivery Activities

Develop

Integrate

Production

Prepare
To simple?
I agree...
How ‘bout...
Develop
(Sprint, Iteration, etc.)
Goals/Budgets
Unit Tests/Trends
Story Acceptance
Build Validation/Regression

Integrate
(Build, Merge, Promote, etc.)
Features
Target Load Profiles
Peak Load Profiles
Weakness Identification

Production

Prepare
(Major Release, Configure, Certify, etc.)
Deployments
Maintenance
Rapid Response
Data Forward

External Integrations
Infrastructure/Tuning
Break Point Identification
Capacity Planning
Better, but I like the puzzle analogy!
...and your team can improve performance on each puzzle piece!
... with just a little work...

every day...

from every one.
Want to know how?
A “Test-Driven” Application Performance Management Model
Huh?
What's that, you ask?
The four T’s stand for...

- Proactive
- Micro & Macro
- Establish Goals
- Update Targets

- Units
- Stories
- Tiers
- Resources
- Goals

- Dev & Prod
- Times
- Resources
- Sizes
- Frequencies
- Dashboard!

- Assess
- Compare
- Investigate
- Accept
- Answer
... and you can apply them like...
... but and is better ...
... and more ands are even better!
Let me help you convince the others...
Value Proposition

- A Few Minutes, Every Day
- Minimizes Surprises
- Methodology / Tool Agnostic
- Adapts to Current Priority or Risk
- Big Visible Dashboards
- No Purchase Necessary
- Simplifies Root-Cause Analysis
- Collaborative
Challenges

- Some Education Required
- Demands Discipline
- Culture Change is Hard
- Several Cycles to Full Value
Footnote 1:
For best results I recommend bundling with...
What is it?

An approach to respond to a specific performance-related question after 4 or fewer hours of team effort with 1 or more of:

A) The answer

B) A partial answer
  - To determine the value of additional effort
  - The level of effort to provide the answer

C) Better questions to address the underlying concern
... and ...
Heuristic Load Testing Throughout the Life-Cycle
Load Testing Principles

**Context**
Project context is central to successful testing.

**Criteria**
Business, project, system, & user success criteria.

**Design**
Identify system usage, and key metrics; plan and design tests.

**Instrument**
Install and prepare environment, tools, & resource monitors.

**Script**
Script the tests as designed.

**Execute**
Run and monitor tests. Validate tests, test data, and results.

**Analyze**
Analyze the data individually and as a cross-functional team.

**Report**
Consolidate and share results, customized by audience.

**Iterate**
"Lather, rinse, repeat" as necessary.

Throughout the Life-Cycle
In other words...
A Practical & Holistic Approach for Achieving Fast & Scalable Systems
Footnote 2:
Unit-Level Testing Tools
(‘cause folks always ask)

FireBenchmarks; Performance testing addin for NUnit
JUnitPerf; a collection of JUnit test decorators for performance
Firefox Performance Tester's Pack
HTTPerf
Footnote 3:

These slides have been given to ignite to post/distribute & they will go on my site shortly.
Preventing Poor Performance with a little work...

every day...

from every one.
Contact Info

Scott Barber
Chief Technologist
PerfTestPlus, Inc

E-mail: sbarber@perftestplus.com
Blog: scott-barber.blogspot.com

Web Site: www.PerfTestPlus.com
Twitter: @sbarber
Review & Questions

Did we learn anything?