Making Sense of the Numbers

DOs and DON'Ts of Quality Performance Testing

Erik Webb, Jeff Beeman
Sr. Technical Consultants, Acquia

Building Bridges, Connecting Communities
About Erik

- Senior Technical Consultant
- Focus on Performance, Infrastructure, and Scalability
- Joined Acquia in early 2010
- 5+ years with Drupal
- 10+ years with LAMP
- Red Hat Certified Engineer
About Jeff

- Senior Technical Consultant
- Focus on Architecture, Deployment, Delivery
- Joined Acquia in late 2010
- 8+ years with Drupal
- Lead architect on several large-scale Acquia PS projects
Everyone needs performance testing.
Do have a plan.

- Creating representative user scenarios
- Scripting the actions of the simulated users
- Analyzing the results of the test
Do involve everyone.

- Not just for developers
- Involve sysadmins, developers, and project managers
- Project stakeholders
- Everyone is the QA team!
- Align tests with business goals
Let’s get on the same page.
Performance

GET drupal.org
Scalability

1s

~1s
Scaling

Vertical

Horizontal
Load Testing

10 users

100 users
Stress Testing

?? users
Contention
Your numbers are only as good as your tests.
Do set technical goals.

- Monitor changes and usage side-by-side
- Ensure new development does not affect existing functionality
- Reduce infrastructure complexity through early testing
- Hold programmers accountable for performance, not just sysadmins
Do set specific success metrics.

- Backend performance
  - X concurrent authenticated sessions
  - X page views per minute
  - X seconds maximum per request
  - X MB memory usage per request
  - Maximum % CPU or GB RAM used on server
- Frontend performance
  - X seconds until initial render
  - X seconds until full page load
Do gather all the data.

- Never trust the client.
- Find the most value for the client.
- Understand the full client use cases.
- Historical data is a requirement.
- We’re too smart to ignore our intuition.
Don’t give up until it’s done.

✓ Get your hands dirty.
✓ Don’t get lost in Drupal.
✓ Assume no one has checked the easy stuff.
✓ Don’t accept anything less than perfect.
Your numbers are only as consistent as your infrastructure.
Don’t use a dev environment for testing.

- Developer overhead - XDebug, XHProf
- Verbose logging
- Congested network
Don’t extrapolate results.
Don’t extrapolate results.

Waiting query #1

Waiting query #2

20 ms lock + query time

Waiting query #3

20 ms lock + prev lock time

Waiting query #4

30 ms lock + prev lock time

Waiting query #5

30 ms lock + prev lock time

Locking query #1

20 ms lock

Locking query #2

20 ms lock + query time

Waiting query #1

20 ms lock + query time

Waiting query #2

20 ms lock + query time

Waiting query #3

30 ms lock + prev lock time

Waiting query #4

30 ms lock + prev lock time

Waiting query #5

30 ms lock + prev lock time

CONTENTION!!!
Your numbers are only as current as your last test.
Do write smart tests.

- User scenario created by business
- Use real data from analytics
- Ensure variability
- Measure test coverage
- New tests for each feature
Don’t write your tests for launch.

- Launch is a milestone, not the end.
- New features added all the time.
- It’s just QA!
Do use CI for performance too.

- Ongoing content scaling complicates long-term reliability
- Integrate with Jenkins (or other CI tool) for performance regression testing
Your numbers are only as good as your tools.
Do multiple types of testing.

- Request profiling
- Service testing
- Simple HTTP response testing
- Load testing
Virtual vs. Real Load Testing

- Virtual
  - HTTP client
  - Designed for efficiency
  - Limited client functionality
  - JMeter cloud service - $500/month up to 4800 concurrent

- Real
  - Browser client
  - Estimates real user experience
  - Supports AJAX natively
  - Selenium-based service - $499/week up to 100 real users
Virtual Users
Real Users
Questions?
We’re Hiring!
@erikwebb
@doogiemac
What did you think?

Evaluate this session at: portland2013.drupal.org/schedule.

Thank you!