Continuous Performance Testing: Challenges and Approaches

Alexander Podelko
Sr. Performance Engineer
Amazon Web Services

1

Alex Podelko

- Has specialized in performance since 1997
- Senior Performance Engineer at AWS – Amazon Aurora
  - Before worked for MongoDB, Oracle/Hyperion, Intel, and Aetna
- CMG Board Director
- SPEC RG Steering Committee Member

Disclaimer: The views expressed here are my personal views only and do not necessarily represent those of my current or previous employers. All brands and trademarks mentioned are the property of their owners. All products are mentioned as examples only, not as recommendations.

2

Adjusting Performance Engineering to Industry Trends

3

Industry Trends

- Web
  - Centralization, open / unlimited workload
- Cloud
  - Further centralization, price tag (FinOps)
  - Dynamic configurations / Self-Management
- Agile / iterative development
  - Continuous Integration / Delivery / Deployment
  - DevOps / SRE

The Past, Present, and Future of Performance Engineering

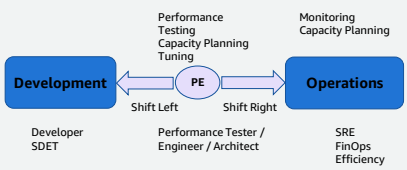
4

All Interconnected

Centralization
=> Control over deployments
=> Ability to deploy small changes
=> Agile development
=> Fuzzier line between Dev and Ops (DevOps, SRE)
=> Need for continuous performance engineering

5

Integrating Performance Engineering into DevOps



Expand or be Squeezed Out ?

6

Adjusting Performance Testing to Agile and CI

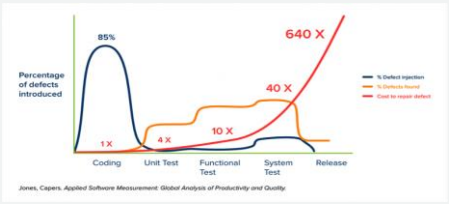
- Agile development should be rather a trivial case for performance testing
  - Working system on each iteration by definition
  - You need performance engineer for the whole project
    - Savings come from detecting problems early
- Addressing deficiencies of the traditional performance testing
  - Early Performance Testing
  - Continuous Performance Testing

7

Early Performance Testing

8

Cost of Fixing Defects Earlier Is Significantly Lower



9

Early Testing - 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"

10

Continuous Performance Testing

11

Integration into Agile and CI/CD

- Continuous performance testing
  - To catch regressions early
- Collecting all info needed to investigate regressions
  - In the form convenient for further analysis
- Foundation to build further automation on the top of it
  - For further performance optimization
- All context-dependent
  - Don't wait for an exact recipe, figure it out depending on your needs

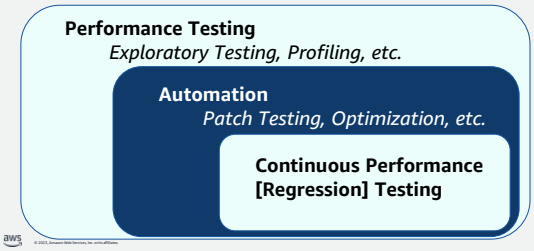
12

Performance Testing Traditional vs Continuous

• Before releases	• Often (maybe even each build)
• Realistic Mix	• Different tests
▪ As close to production as possible	▪ To maximize coverage
• Checking Service Level Objectives (SLOs)	• Checking the difference between builds
• Using a load testing tool or harness	• Using an additional layer of automation on the top of load testing tool
• The approach is relatively consistent and well described	• All context-dependent

13

My View / Use of Terminology



14

Challenges of Continuous Performance Testing

- Integration
- Coverage Optimization
- Variability / Noise Reduction
- Change Detection
- Advanced Analysis
- Operations / Maintenance

15

The Challenge of Integration

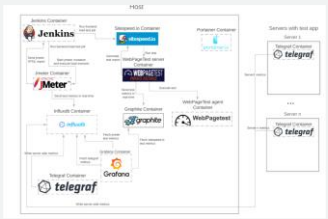
16

Continuous Integration: Load Testing Tools

- CI support in load testing tools
  - Integration with CI Servers (Jenkins, Hudson, etc.)
  - Automation support
- CI tools support for performance testing
  - Jenkins Performance Plugin
- Performance Testing Frameworks
  - Combining multiple tools

17

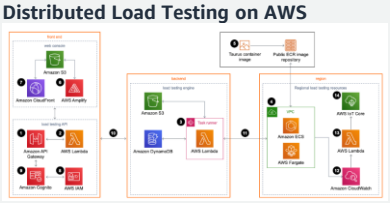
A Performance Testing Framework



An example:
https://github.com/serputko/performance-testing-framework

18

Distributed Load Testing on AWS



From AWS Solutions Library
https://aws.amazon.com/solutions/implementations/distributed-load-testing-on-aws/

19

Closely Integrated Systems

- Sophisticated, but proprietary closely integrated systems
  - Creating a Virtuous Cycle in Performance Testing at MongoDB
  - Fallout: Distributed Systems Testing as a Service (DataStax)
  - Tracking Performance of the Graal Compiler on Public Benchmarks (Charles University / Oracle Labs)
  - Introducing Ballast: An Adaptive Load Test Framework (Uber)

20

Unified Development Services

Such as AWS CodeCatalyst
https://aws.amazon.com/codecatalyst/

21

The Challenge of Coverage Optimization

22

Time / Resource Considerations

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

23

Coverage Optimization

- A multi-dimensional problem
  - Configuration
  - Workloads / Tests
  - Frequency of runs
- A trade off between coverage and costs
  - Costs of running, analyzing, maintenance, etc.

24

The Challenge

- If addressed seriously, the number of workloads / tests / configurations is growing
- No good way to optimize
- One approach is to see if some results are correlated
  - If we find same problems on the same set of tests, we can use just one or few tests from this group
  - Tracking Performance of the Graal Compiler on Public Benchmarks (Charles University / Oracle Labs)
- Combinatorial testing approaches (PairWise / Covering Arrays)
  - From functional testing

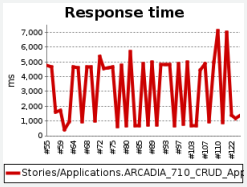
25

The Challenge of Variability

26

Variability - Environment

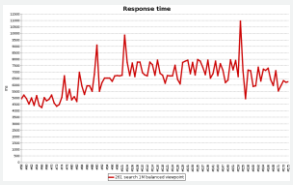
- Due to difference in environments



27

Variability - System

- Inherent to the test setup



28

Addressing Variability

- Methodological principles for reproducible performance evaluation in cloud computing. 2019 (SPEC RG – Cloud)
- Reducing variability in performance tests on EC2: Setup and Key Results (MongoDB)
- Tracking Performance of the Graal Compiler on Public Benchmarks

29

Addressing Variability

- Same environment / starting config
  - For example, AWS cluster placement groups
- No other load
- Multiple iterations
- Reproducible multi-user tests
  - Restarts between tests
  - Clearing caches / Warming up caches
  - Staggering / Sync points

30

The Challenge of Change Detection

31

Complex Results

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

32

Simple Comparison

Jenkins Performance Plugin

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 viewpoint	1	0	81126	723

33

keptn.sh

```
---
spec_version: "1.0"
comparison:
  aggregate_function: "avg"
  compare_with: "single_result"
  include_result_with_score: "pass"
  number_of_comparison_results: 1
filter:
objectives:
  - sli: "response_time_p95"
    key_sli: false
    pass:             # pass if (relative change <= 10% AND absolute value is < 600ms)
      - criteria:
          - "<=+10%"  # relative values require a prefixed sign (plus or minus)
          - "<600"    # absolute values only require a logical operator
    warning:          # if the response time is below 800ms, the result should be a warning
      - criteria:
          - "<=800"
    weight: 1
total_score:
  pass: "90%"
  warning: "75%"
```

Quality Gates
SLIs / SLOs as code

34

Change Point Detection

- Statistical methods taking noise in consideration
- E-Divisive means algorithm
  - ICPE Paper: Change Point Detection in Software Performance Testing
  - Fixing Performance Regressions Before they Happen, Netflix Technology Blog
  - https://github.com/mongodb/signal-processing-algorithms
    – Open sourced, generic
  - Need several data points. May miss a gradual degradation.

35

The Challenge of Advanced Analysis

36

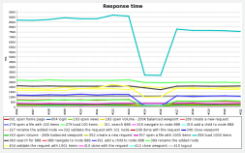
Keep All Artifacts for Further Analysis

- Get all metrics
  - Throughputs, latencies, resource utilizations, etc.
- Save all related artifacts
  - Exact code / configuration
  - Logs, etc.
- Ability to re-run the test in the exactly same configuration is helpful

37

Visualization

- Visualizing systems and software performance - Report on the GI-Dagstuhl
- Sometimes helps to catch an issue



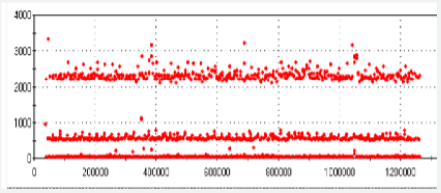
38

Looking Beyond Aggregate Info

39

Looking at Individual Results Patterns

Scatter charts – a "banding" pattern from
http://www.perftestplus.com/resources/BPT6.pdf



40

The Challenge of Operations and Maintenance

41

Operations

- Scheduling / execution tests
- Results analysis
- Triaging and escalating issues
- Maintenance

42

Coverage / Maintenance Trade-Off

43

Catching / Troubleshooting Errors

- Catching errors is not trivial
  - Building in checks
  - Depends on interfaces used
    - Protocol-level [recording]
    - GUI
    - API/Programming
    - Production Workloads
- Keeping logs / all info needed to investigate issues

44

Changing Interfaces

- If using protocol-level or GUI scripts, minor changes may break them
  - It may be not evident
  - If recording used, a change in interfaces may require to recreate the whole script
- API / Programming is usually more stable / easier to fix
- AI to catch the changes / self-healing scripts

45

Who Is Doing Maintenance?

- Who is responsible for what?
- Infrastructure Code
  - Tools, plumbing code, integration
- Specific tests
- Integrated workloads
  - Covered multiple functional areas

46

Performance Skills Trends

47

Skills in Demand

- All old good performance knowledge / skills
  - Not as much around load testing tools anymore
- Development / Scripting / Automation
  - Needed for early / continuous testing
- Performance understanding becoming a must in the industry
  - Need to go one level deeper

48

Algorithmic Complexity

- Time Complexity
- Space Complexity
- Big-O notation

Almost in every interview around the globe !

- Connect it with practical performance engineering?

49

[System Design Interview Cheat Sheet](#)
by Vahid Dejwakh

Just ***an example*** of the changing attitude

50

AWS Well-Architected Framework

[The 6 Pillars of the AWS Well-Architected Framework](#)

- Operational Excellence
- Security
- **Reliability**
- **Performance Efficiency**
- **Cost Optimization**
- Sustainability

51

Less Attention to Load Testing Tools

- Performance engineering shifted to
  - Other ways to mitigate performance risk
  - More closely integrated continuous performance testing
- Proliferation of APIs / simple open-source tools

52

SUMMARY

- Adjusting Performance Testing to Agile and CI is the main trend
- Specific challenges should be addressed:
  - Integration
  - Coverage Optimization
  - Variability / Noise Reduction
  - Change Detection
  - Advanced Analysis
  - Operations / Maintenance
- Performance engineering gets more integrated, context-dependent
  - Integrated into both **Dev**elopment and **Op**eration**s**

53

Thank you!

Alex Podelko
podealex@amazon.com

54