Performance Testing in a Containerized World

Paola Rossaro
Something about me

- PhD Computer Science (performance)
- 20+ years high-tech
- CTO and Co-founder Nouvola
- And a unicorn mom!
Agenda

• Continuous Testing
• Performance testing
• Containers and Microservices
• Containers vs. non-containers performance
• Some examples
• Key Takeaways
What you are going to learn

- Continuous Integration and performance testing
- Best practices for performance testing of microservices and containers
- How containers can help performance
- What performance metrics you can derive
Typical Issues

• **Speed and responsiveness**
  – Application doesn’t live up to users expectations for speed, responsiveness and uptime

• **Release cycles**
  – Current release cycle processes are not meeting market demands

• **QoS**
  – Quality of service issues are jeopardizing brand and customer loyalty

• **Application health**
  – Visibility of application health is reactive vs. proactive (i.e. break/fix)

• **Development efficiency**
  – Development and test resources spend time on non core business activities
New solutions

• DevOps / Agile flows
• Continuous Integration and Deployment
• Continuous Testing
• Containers and microservices
• Cloud based and Serverless based applications
• ...

monolith  n-tier  service oriented  microservices  serverless
Performance Testing
Legacy solutions won’t work any longer

- Not a good fit for DevOps / Agile flows
- Cumbersome, complex products do not appeal to modern developer
- Hinder transitions to DevOps and acceleration of release cycles

Simple load testing is not enough

- Need real world scenarios to have credible data
- Need actionable insights to iterate and optimize
- Open source tools require hundred of hours of dev time
What is DevOps?

Ambiguity about what DevOps means

Multiple reference architectures

“Cloudy with a chance of DevOps”

http://www.slideshare.net/SonatypeCorp/nexus-and-continuous-delivery
http://bridgetkromhout.com/speaking/2016/devopsdays-london/
DevOps Pipeline

- **Delivery team**
  - Check in
  - Feedback
- **Version control**
  - Check in
  - Trigger
  - Feedback
- **Build & unit tests**
  - Trigger
  - Feedback
  - Trigger
  - Feedback
- **Automated acceptance tests**
  - Trigger
- **User acceptance tests**
  - Approval
- **Release**
  - Approval
Continuous testing

“Continuous Integration and Delivery are (almost) meaningless without Continuous Testing.”

https://sdarchitect.wordpress.com/2012/10/30/understanding-devops-part-4-continuous-testing-and-continuous-monitoring/
Development env

Integration testing

Functional testing

Test env

Performance + acceptance testing

Preproduction env

Continuous Delivery

Continuous testing

Production env

Monitoring

nouvola
Why Containers

Containers advantages

• Modular
• Easy change processes
• Isolation from code issues of other containers/services
• Cleaner interactions with other components
Why Containers and Microservices

Containers are the natural “home” for microservices:

• Modular paradigm fits with microservices
• Can have more than one service per VM
• Can be easily moved on different environments
• Can easily have versions of microservices on same VM and switch
Containers and Microservices Performance

- Measure in isolation
- Measure dependencies
- Compare different environments and configurations
Containers/Microservices Performance

TEST

Instantiate
Millions of Users

Containerized app(s) or microservice(s)

MEASURE

Optimize
And Improve

Very easy for developers to understand immediately how their code is performing in the REAL WORLD
Best Practices

- Basic Testing
- Scalability Testing
- Resiliency Testing

3 Types of testing => SUCCESS!
Best Practices

• Test in isolation
• Test the whole system
• AUTOMATE!
Best Practices

- Create workflow for functional testing
- Run functional test
- Modify as you see fit for load testing
- Expand to multiple services/containers
- Check results: load + resiliency
Don’t forget Docker

• Fast Deployment
• Different versions can be deployed
• Very quick to switch/reset Environments
EXAMPLES: QA-Dashboard
QA-Dashboard

- Testing Dashboard being built on microservices
- Spinning up containers for Services for test cases
- Performance testing environments costs smaller with containers
QA-Dashboard Recipe

• Postman for functional testing
• Import into Nouvola DiveCloud
• Modify as you see fit
• Combine multiple scenarios
• Check results: load + resiliency
Performance TESTING and ANALYTICS for WEB Applications, Mobile and API

**Turnkey**
Automated and easy.
Self-service.
Developers love it.

**DevOps Ready**
Best Fit with Agile, DevOps and modern development environments

**Integrated Analytics**
Powerful integrated analytics. Actionable.
Lead to insights
DevOps + Performance Testing

Integration
- Jenkins Plugin

Delivery
- AWS Code Pipeline

Operations
- New Relic

API/Ruby Gem
First: Test in isolation

- Create a Postman workflow
- Test functionality
- Export workflow to Nouvola
- Create and Run a performance test
- Check results
Test in isolation: minimal set
Test in isolation: Postman workflow
Test in isolation: export
Test in isolation: import to Nouvola

You can also record a session with DiveProxy. [USE DIVEPROXY]

<table>
<thead>
<tr>
<th>ID</th>
<th>DATE</th>
<th>NAME</th>
<th>SAVED VIA</th>
</tr>
</thead>
<tbody>
<tr>
<td>4174</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4173</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4169</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4136</td>
<td>07/10/17 at 3:21:19 pm</td>
<td>cvtest</td>
<td>Postman</td>
</tr>
<tr>
<td>4129</td>
<td>07/10/17 at 3:21:19 pm</td>
<td>SSD</td>
<td>Postman</td>
</tr>
</tbody>
</table>

[Import type]
- DiveCloud session export file
- Postman file
- Selenium script file
Test in isolation: Parameterize

Sessions > DiveCloud Test > Request

http://192.241.237.188:1234/testplans

- Show instructions for random requests

**HEADERS**

**REQUEST DATA**

- Show instructions for random requests

<table>
<thead>
<tr>
<th>NAME</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Not set]</td>
<td>{ &quot;name&quot;:&quot;DiveCloud Test $$SS$$&quot;, &quot;author&quot;: &quot;%name%&quot; }</td>
</tr>
</tbody>
</table>

**SESSION VARIABLES**

- Show instructions for session variables

<table>
<thead>
<tr>
<th>NAME (LETTERS ONLY)</th>
<th>VALUE</th>
<th>TYPE</th>
<th>FILE-BASED VARIABLE SET</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>[Not set]</td>
<td>Static</td>
<td>test_users_ex.csv</td>
</tr>
</tbody>
</table>

Choose File: No file chosen
Randomize selection: No
Test in isolation: Run a test

<table>
<thead>
<tr>
<th>Plan Duration</th>
<th>minutes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traffic Load</td>
<td>concurrent users</td>
</tr>
</tbody>
</table>

- Use Linear Ramp to increase load over test. How does Linear Ramp work?
- Use Think Time to simulate user latency. How does Think Time work?

Think Time: Do not use Think Time

Think Time per session:

Enter new URLs to test, or use a previously saved test session, including a DiveTrace recorded session. What is a DiveTrace session?

Session URLS:

or: DiveCloud Test

Set the response time threshold you want to use to determine if the plan will pass. What is a good threshold value?

Response threshold: 600 msec

Show advanced options

Select the geographic regions and percentages you want to include as part of your plan or select Map View (Return to Dropdown)

Plan: Nuvola Micro Service Demo
ID: 5029
Status: Not Started
Results: 0
Description: POST /Testplans
Test in isolation: Check Results
Test multiple services

- Microservices architecture makes it easier to add and combine services to the testing effort.
- Nouvola Multi-session makes it easier to combine multiple services together.
- Test for load and resiliency subsystem or entire systems.
Test multiple services
Test multiple services

New Multi Session

Enter name

Name: POST and GET

Choose multi-session type

Combined Session Type: Sequential

"Sequential" sessions are replayed one after another in sequence, while "parallel" sessions are replayed at the same time.

Enter order

- DiveCloud Test
- DiveCloud Test GET

Order:
1. DiveCloud Test
2. DiveCloud Test GET

SAVE
Optimize Docker settings

- Change CPU share
- Change memory
- Test, measure, tweak, iterate

#### Isolation

<table>
<thead>
<tr>
<th>CPU</th>
<th>Shares 1024 shares</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Allowed Cores 0 1</td>
</tr>
</tbody>
</table>

| Memory | Reservation unlimited | Limit unlimited |
Compare with/without Docker

- Setup with containers after optimization
- Execute microservices scenarios without Docker
- Execute microservices scenarios with Docker
Compare with/without Docker

Traffic Load: 2  Threshold: 600 msec  Region: Northeast North America

COMPARISON  TEST #4436  TEST #4435

AVERAGE RESPONSE TIME PER SELECTED TESTS

Test Duration (sec)

Page Load (msec)

# of Concurrent Users

Test #4436  Test #4435
Conclusions

• Microservices architecture makes it easier to do performance testing upfront

• Containerization speeds up performance Testing lifecycle

• With proper knowledge and planning, performance results on containers should be on par or better than without
Thank You

paola.rossaro@nouvola.com
Sign up for free at www.nouvola.com