Integrating Automated Scalability Assessment into DevOps

Alberto Avritzer et al.
About Alberto Avritzer

eSulabSolutions

• Senior Researcher at Siemens and AT&T Bell Labs for about 24 years
• Published over 70 papers in journals, refereed conference proceedings, and book chapters in those areas: (http://dblp.uni-trier.de/pers/hd/a/Avritzer:Alberto)
• Siemens/MTA: The NY Subway PA/CIS Scalability Assessment
• Load Testing and Performance Analysis of AT&T Operations support systems:
  • Monitoring for Software aging and Rejuvenation (1993)
  • Performance testing using Markov chain (1995)
• Founder of eSulabSolutions (https://esulabsolutions.godaddysites.com/):
  • Automated scalability assessment in DevOps and micro service architecture
Motivation: Recent Scalability Related Disasters

Americans crash the Canadian immigration

NYS Labor website crashes
Motivation: Influence of Poor Performance on the Success of Businesses

- 50% of the online customers leave the website after load times > 2 seconds.
- Only 40% of the customers return after having experienced a performance problem.

Immediate response:
- Users notice the delay, sense of flow gets lost

Users' attention gets lost:
- Response time (log 10)
  - 0.1 s
  - 1.0 s
  - 10.0 s

Users need to reorient themselves before each interaction
Examples of Operational Profile Representations

Observed load situations

Empirical distribution of load situations

<table>
<thead>
<tr>
<th>Request</th>
<th>Orig.</th>
<th>Rel.</th>
</tr>
</thead>
<tbody>
<tr>
<td>add to cart</td>
<td>63,761</td>
<td>0.07</td>
</tr>
<tr>
<td>cancel order</td>
<td>632</td>
<td>0.00</td>
</tr>
<tr>
<td>clear cart</td>
<td>6,047</td>
<td>0.01</td>
</tr>
<tr>
<td>defer order</td>
<td>6,782</td>
<td>0.01</td>
</tr>
<tr>
<td>home</td>
<td>59,934</td>
<td>0.07</td>
</tr>
<tr>
<td>inventory</td>
<td>30,596</td>
<td>0.03</td>
</tr>
<tr>
<td>login</td>
<td>61,500</td>
<td>0.07</td>
</tr>
<tr>
<td>logout</td>
<td>59,934</td>
<td>0.07</td>
</tr>
<tr>
<td>purchase cart</td>
<td>8,360</td>
<td>0.01</td>
</tr>
<tr>
<td>remove</td>
<td>3,027</td>
<td>0.00</td>
</tr>
<tr>
<td>sell inventory</td>
<td>66,679</td>
<td>0.08</td>
</tr>
<tr>
<td>shopping cart</td>
<td>9,074</td>
<td>0.01</td>
</tr>
<tr>
<td>view items</td>
<td>498,601</td>
<td>0.57</td>
</tr>
<tr>
<td><strong>Σ</strong></td>
<td>874,927</td>
<td>1.00</td>
</tr>
</tbody>
</table>
DevOps is...

...a set of Practices

intended to reduce the time between committing a change to a system and the change being placed into normal production,

while ensuring HIGH QUALITY.

COST-EFFICIENT
RELIALBE
SECURE
FAST
RESILIENT
ELASTIC
CD Pipelines...
Automation
Microservices
Agile
*-as-Code
Docker
Chaos Engineering
Polyglot
Live Experimentation

Avritzer et al.: Automated Scalability Assessment in DevOps
My Questions of Interest

How can DevOps practices support quality assurance?

How to integrate "established" quality assurance with DevOps practices?
- PPTAM: Production and Performance Testing Based Application Monitoring
Scalability Requirement Automated Measurement

Avritzer et al.: Automated Scalability Assessment in DevOps
The Classic Load Testing Approach

... and Classic Problems

- High manual effort for maintaining load tests
- There are no suitable load tests
- Load tests need much time to execute
- Complex analysis of performance regressions

**Workload Specification**

**Test Results**

**Evaluation**

- Expert
- Load Driver
- Expert/Basic Rules
Load Testing in Continuous Delivery Pipelines … How Problems Get Worse

High manual effort for maintaining load tests vs. Pipeline automation

Load tests need much time to execute vs. Fast & frequent releases

Service-focus requires multiple tests vs. There are no suitable load tests

Complex load tests for every release impossible vs. Complex analysis of performance regressions

void main(String[] args) {
    int foo;
    // do something
    bar(foo);
    System.out.println("Hi");
}

Implementation Build Functional Testing Performance
• PPTAM: Production and Performance Testing Based Application Monitoring
Quantitative Assessment of Deployment Alternatives

• Challenge: assess performance of architectural deployment alternatives (e.g., number of replicas, CPU/memory allocation, technology stack) under fuzzy requirements

• Our approach
  • Use operational data to generate and weigh load tests
  • Measure baseline requirements
  • Metric allows quantitative comparison of deployment alternatives
  • Builds on previous work from telecommunication systems:

Alberto Avritzer, Vincenzo Ferme, Andrea Janes, Barbara Russo, Henning Schulz, and André van Hoorn:
*A Quantitative Approach for the Assessment of Microservice Architecture Deployment Alternatives by Automated Performance Testing.*

Overview of Approach

1. Observed load situations over time.
2. Empirical distribution of load situations.
3. Test results: 0.12, 0.14, 0.20, 0.16, 0.11. Test outcomes: ✔️ ✔️ ✗ ✗.
4. Domain Metric: 0.73
   - Scalability criteria
   - Baseline test
   - Deployment conf.

Avritzer et al.: Automated Scalability Assessment in DevOps
Experiments

12 microservices
Production

6 Load Levels
50, 100, 150, 200, 250, 300 Concurrent user sessions
Empirical Distribution of Load situations
Sampled Load Tests
Custom Op. Mix

1, 2

10 configurations
RAM
CPU
Replicas
Deployment Config.

Scal = avg + 3σ
Experiment Results: Computation of Domain Metric (1/2)

<table>
<thead>
<tr>
<th>API</th>
<th>Scalability Criteria</th>
<th>Users</th>
<th>Aggr. Rel. Freq.</th>
<th>Contrib. to Domain Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET /</td>
<td>PASS</td>
<td>50</td>
<td>0.10582</td>
<td></td>
</tr>
<tr>
<td>GET /cart</td>
<td>PASS</td>
<td>100</td>
<td>0.18519</td>
<td></td>
</tr>
<tr>
<td>POST /item</td>
<td>FAIL</td>
<td>250</td>
<td>0.20370</td>
<td>Max: 0.20370</td>
</tr>
<tr>
<td></td>
<td></td>
<td>300</td>
<td>0.06085</td>
<td>Actual: 0.13580</td>
</tr>
</tbody>
</table>

Deployment Configuration: 1 GB RAM, 0.25 CPU, 1 Replica
## Experiment Results: Computation of Domain Metric (2/2)

### Table: Users vs. Contribution

<table>
<thead>
<tr>
<th>Users</th>
<th>Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>0.10582</td>
</tr>
<tr>
<td>100</td>
<td>0.18519</td>
</tr>
<tr>
<td>150</td>
<td>0.22222</td>
</tr>
<tr>
<td>200</td>
<td>0.07999</td>
</tr>
<tr>
<td>250</td>
<td>0.13580</td>
</tr>
<tr>
<td>300</td>
<td>0.04729</td>
</tr>
</tbody>
</table>

### Contributors to Domain Metric

Deployment Configuration: 1 GB RAM, 0.25 CPU, 1 Replica

Max: 1

Actual: 0.77631
## Experiment Results: Single-Metric Comparison of Alternatives

<table>
<thead>
<tr>
<th>RAM</th>
<th>CPU #</th>
<th>Cart Replicas</th>
<th>Domain Metric (HPI)</th>
<th>Domain Metric (FUB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5 GB</td>
<td>0.25</td>
<td>1</td>
<td>0.61499</td>
<td>0.54134</td>
</tr>
<tr>
<td><strong>1 GB</strong></td>
<td>0.25</td>
<td>1</td>
<td><strong>0.77631</strong></td>
<td><strong>0.53884</strong></td>
</tr>
<tr>
<td>1 GB</td>
<td>0.5</td>
<td>1</td>
<td>0.53559</td>
<td>0.54106</td>
</tr>
<tr>
<td>0.5 GB</td>
<td>0.5</td>
<td>1</td>
<td>0.51536</td>
<td>0.54773</td>
</tr>
<tr>
<td>0.5 GB</td>
<td>0.5</td>
<td>2</td>
<td>0.50995</td>
<td>0.54111</td>
</tr>
<tr>
<td>1 GB</td>
<td>0.25</td>
<td>2</td>
<td>0.74080</td>
<td>0.54785</td>
</tr>
<tr>
<td>1 GB</td>
<td>0.5</td>
<td>2</td>
<td>0.53401</td>
<td>0.54106</td>
</tr>
<tr>
<td><strong>0.5 GB</strong></td>
<td>0.5</td>
<td>4</td>
<td>0.50531</td>
<td><strong>0.54939</strong></td>
</tr>
<tr>
<td><strong>1 GB</strong></td>
<td>0.25</td>
<td>4</td>
<td><strong>0.37162</strong></td>
<td><strong>0.54272</strong></td>
</tr>
<tr>
<td>1 GB</td>
<td>0.5</td>
<td>4</td>
<td>0.56718</td>
<td>0.54271</td>
</tr>
</tbody>
</table>
Experiment Results: Visual Comparison of Alternatives

Contrib. to Domain Metric vs. Sampled Load Tests

Max Contrib.
Depl. Conf.

Avritzer et al.: Automated Scalability Assessment in DevOps
PPTAM

- Production and Performance Testing Based Application Monitoring
Avritzer and Russo: Operational Profile Data for Continuous Dependability Assessment in DevOps
Mirai BotNet

- Mirai is a malware that has been used to turn networked devices running Linux into remotely controlled bots
- We use it to attack the system. It can perform different types of attack
- by now, we have explored http, syn, ack
PPTAM, No attack and attack plots with 50 users
Identify Scalability Impacting Architecture Components

- Apply the approach to a large telecom system
Identify Scalability Impacting Architecture Components from Performance Testing Log

<table>
<thead>
<tr>
<th>Counter_name</th>
<th>Measurement</th>
<th>Load</th>
<th>Service_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response Time_X</td>
<td>Y</td>
<td>L</td>
<td>S</td>
</tr>
<tr>
<td>Response Time_X</td>
<td>Y</td>
<td>L</td>
<td>S</td>
</tr>
<tr>
<td>Response Time_X</td>
<td>Y</td>
<td>L</td>
<td>S</td>
</tr>
<tr>
<td>Response Time_X</td>
<td>Y</td>
<td>L</td>
<td>S</td>
</tr>
</tbody>
</table>

What is the problem with this log?
Compute Scalability Baseline, Normalized Distance from Baseline, and Linear Regression Slope

<table>
<thead>
<tr>
<th>Counter_Name</th>
<th>Low Load_RT</th>
<th>Std_RT</th>
<th>Baseline_RT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response Time X</td>
<td>2.488602</td>
<td>3.513397</td>
<td>12.033352</td>
</tr>
</tbody>
</table>

![Graph showing normalized distance vs. performance trend]
## Compute Scalability Baseline, Normalized Distance from Baseline, and Linear Regression Slope

<table>
<thead>
<tr>
<th>Counter_Name</th>
<th>Low Load_ RT</th>
<th>Std _RT</th>
<th>Baseline_RT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response Time X</td>
<td>2.488602</td>
<td>3.513397</td>
<td>12.033352</td>
</tr>
</tbody>
</table>

### Linear Regression

<table>
<thead>
<tr>
<th>counter_name</th>
<th>ndistance</th>
<th>slope</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adjustment</td>
<td>0.3702358335</td>
<td>-1.6849765621</td>
</tr>
<tr>
<td>Enquiry</td>
<td>1.237823329836572</td>
<td>0.32741495692</td>
</tr>
<tr>
<td>Interrogation</td>
<td>1.1421493733</td>
<td>0.38719590263</td>
</tr>
<tr>
<td>Resources Read</td>
<td>0.9151258042</td>
<td>0.3500825151</td>
</tr>
<tr>
<td>Resources Update</td>
<td>0.8475972966</td>
<td>1.4582236379</td>
</tr>
<tr>
<td>Status Updates</td>
<td>1.5746310526</td>
<td>35.946471084</td>
</tr>
<tr>
<td>Control</td>
<td>1.2888398316</td>
<td>0.3901288114</td>
</tr>
<tr>
<td>DB Data Management</td>
<td>1.4307154532</td>
<td>1.263215473</td>
</tr>
<tr>
<td>Internal Communication</td>
<td>0.8258893410</td>
<td>0.1369599249</td>
</tr>
<tr>
<td>Offline</td>
<td>1.7349712818</td>
<td>0.0012880585</td>
</tr>
<tr>
<td>Online</td>
<td>0.9393607853</td>
<td>0.9230591413</td>
</tr>
<tr>
<td>Recompose</td>
<td>1.2213424785</td>
<td>1.3542574187</td>
</tr>
</tbody>
</table>
Multivariate analysis used to prioritize re-factoring using Slope and Normalized Distance Profile
Summary

- DevOps provides access to real-time data, enables production and testing integration
- Use cases
  - Performance, Scalability, Security
  - other illities: survivability, resilience
- Provide detailed guidance to architects and developers in real time:
  - Domain-metric based scalability assessment dashboards
  - Prioritize component re-factoring based on load testing results
Publications

