Model-Based Load Testing for Performance and Security

Cornel Barna  Mark Shtern  Michael Smit  Marin Litoiu

Computer Science and Engineering
York University

April 17, 2012
Performance Testing
The Performance Stress Space
The Framework
Results
Search Complexity

DoS Attack Mitigation
DoS Attacks
Adaptive DoS Mitigation
Experiments
Results
Performance Testing
    The Performance Stress Space
    The Framework
    Results
    Search Complexity

DoS Attack Mitigation
    DoS Attacks
    Adaptive DoS Mitigation
    Experiments
    Results
Introduction
The Performance Stress Space

Consider a system with two *hardware* resources, two *software* resources and two *classes of service*.

For the hardware resources, the constraints are given by:

\[ U^h_K = \sum_{\forall C \in C} \frac{D_{K,C}}{D_{K_r,C}} \times U_{K_r,C}, \quad \forall K \in K^h \]

With feasible space \( \rightarrow OABC \).

**Hardware Constraints**

- Linear equations:
- Feasible space \( \rightarrow OABC \)

**Software Constraints**

- Non-linear equations:
- Feasible space \( \rightarrow OABDE \)
The Performance Stress Space

Consider a system with two *hardware* resources, two *software* resources and two classes of service.

\[ U_h^1 = a \quad U_h^2 = b \quad U_s^1 = c \quad U_s^2 = d \]

**Hardware Constraints**

- **Linear equations:**
  \[ U_K^h = \sum_{\forall C \in C} \frac{D_{K,C}}{D_{K_r,C}} \times U_{K_r,C}, \quad \forall K \in K^h \]
  
  - Feasible space \( \rightarrow \) \( OABC \)

**Hardware and Software Constraints**

- **Non-linear equations:**
  \[ U_K^s = \sum_{\forall C \in C} \frac{R_{K,C}}{D_{K_r,C}} \times U_{K_r,C}, \quad \forall K \in K^s \]
  
  - Feasible space \( \rightarrow \) \( OABDE \)
The Framework

Stress goals

Autonomic Test Controller

Performance Model  Kalman Filter

Test Cases

Component/System under test

Monitor and Estimator

Model loop

Work Generator loop

Model-Based Load Testing
The Framework

- Autonomic Test Controller
- Test Cases
- Component/System under test
- Monitor and Estimator
- Performance Model
- Kalman Filter

Model loop

Work Generator loop

Stress goals
The Framework

Stress goals

Autonomic Test Controller

Performance Model

Kalman Filter

Test Cases

Component/System under test

Monitor and Estimator

Model loop

Work Generator loop

Model-Based Load Testing
The Framework

Stress goals

Autonomic Test Controller

Performance Model

Kalman Filter

Test Cases

Component/System under test

Monitor and Estimator

Model loop

Work Generator loop

Model-Based Load Testing
Results (response time goal)
Results (response time goal)

Users in scenario:
- browse
- buy

Response Time (browse):
- 0
- 20
- 40
- 60
- 80
- 100

Users in scenario:
- browse
- buy

Response Time (buy):
- 40
- 60
- 80
- 100
- 120
- 140
- 160
- 180
- 200
- 220

Cornel Barna, Mark Shtern, Michael Smit, Marin Litoiu
Model-Based Load Testing
Results (response time goal)

Users in scenario buy
Users in scenario browse
Response Time (browse)

Response Time
0 20 40 60 80 100

40 60 80 100 120 140 160 180 200 220

Users in scenario browse
Users in scenario buy

Cornel Barna, Mark Shtern, Michael Smit, Marin Litoiu  
Model-Based Load Testing
Results (response time goal)

Users in scenario

<table>
<thead>
<tr>
<th>Response Time (browse)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>20</td>
</tr>
<tr>
<td>40</td>
</tr>
<tr>
<td>60</td>
</tr>
<tr>
<td>80</td>
</tr>
<tr>
<td>100</td>
</tr>
</tbody>
</table>

Users in scenario

<table>
<thead>
<tr>
<th>Response Time (browse)</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
</tr>
<tr>
<td>60</td>
</tr>
<tr>
<td>80</td>
</tr>
<tr>
<td>100</td>
</tr>
<tr>
<td>120</td>
</tr>
<tr>
<td>140</td>
</tr>
<tr>
<td>160</td>
</tr>
<tr>
<td>180</td>
</tr>
<tr>
<td>200</td>
</tr>
</tbody>
</table>

Cornel Barna, Mark Shtern, Michael Smit, Marin Litoiu

Model-Based Load Testing
Complexity

Exhaustive search:

\[
\begin{align*}
6 \text{ classes of service} & \quad \Rightarrow \quad 98\,619\,368\,490 \text{ workload mixes} \\
\text{at most 200 users} & \quad \Rightarrow \quad 119 \text{ workload mixes}
\end{align*}
\]

With our framework:

\[
\begin{align*}
6 \text{ classes of service} & \quad \Rightarrow \quad 119 \text{ workload mixes} \\
\text{no limit for users} & \quad \Rightarrow \quad 119 \text{ workload mixes}
\end{align*}
\]
Performance Testing
  The Performance Stress Space
  The Framework
  Results
  Search Complexity

DoS Attack Mitigation
  DoS Attacks
  Adaptive DoS Mitigation
  Experiments
  Results
DoS Attacks

- Flood the target server with requests in an attempt to saturate the bottlenecks;
- DDoS attacks: the requests can come from multiple machines;
- Challenging to defend against them;
- Methods of mitigation: firewalls and intrusion detection systems.
Adaptive DoS Mitigation

Performance Monitoring Data (feedback)

Incoming traffic

Reverse Proxy

Decision Engine

Dynamic Firewall

Regular traffic

Analyzer

Outgoing traffic

Filtered traffic

Challenge

Response

Regular traffic

Web Application
Decision Engine

- Use OPERA model to analyze the performance metrics;
- Constantly synchronize the model with the real system;
- When system is overloaded:
  - Identify the scenario(s) that should be filtered;
  - Create filtering rules;
- When system is functioning normally:
  - Identify the scenario(s) that should be unfiltered (if any);
  - Remove filtering rules;
Model-based Protection

Performance goals

Incoming requests

Decision Controller
Request Filter Rules
System under protection

Performance Model
Kalman Filter
Monitor and Estimator

Decision Engine loop
Protection loop

Cornel Barna, Mark Shtern, Michael Smit, Marin Litoiu
Model-Based Load Testing
Web Application with 6 scenarios: marketing, product selection, buy, pay, inventory, auto bundles.
Results without a model

Cornel Barna, Mark Shtern, Michael Smit, Marin Litoiu
Results with OPERA
References


