The Fourth International Workshop on Large-Scale Testing (LT 2015)

Large-Scale Testing: Load Generation

Alexander Podelko
alex.podelko@oracle.com
alexanderpodelko.com/blog
@apodelko

February 1, 2015
About Me

• Have specialized in performance for the last 17 years
• Currently performance testing and optimization of Hyperion products at Oracle
• Board director at CMG (http://cmg.org), organization of performance and capacity professionals
  – Next conference November 2-5, 2015 in San Antonio, TX

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.
Load Testing Process

1. Collect Requirements
2. Define Load
3. Create Test Assets
4. Run Tests
5. Analyze Results
   - Goals are met
   - Goals are not met
     - Modify System

Done
Challenges of LT

• How can we create load?
  – Workload generation

• What load do we want to generate?
  – Test design
Manual

• Not an option for a large number of users
• Always variation in human input times
• Can be a good option to simulate quickly a few users
• Can be used with other methods to verify correctness
Record and Playback: Protocol Level

Load Generator

Load Testing Tool

Virtual Users

Network

Application

Server
Considerations

• Usually doesn't work for testing components
• Each tool support a limited number of technologies (protocols)
• Some technologies are very time-consuming
• Workload validity in case of sophisticated logic on the client side is not guaranteed
• Client-side timing is not included
Record and Playback: UI Level

Load Generator

Load Testing Tool

Virtual Users

Browsers

Network

Server

Application
Different Approaches

• Traditional tools, fat clients
  – Require a separate machine (or a terminal session) per user
• Low-level graphical protocols
  – Citrix, Remote Desktop
• Web tools, browser
  – Require a separate browser instance
• Web tools, light-weight browser
  – Require a separate light-weight browser instance
  – For example, HtmlUnit or PhantomJS
Considerations

• Scalability
  – Still require more resources
• Supported technologies
• Timing accuracy
• Playback accuracy
  – For example, for HtmlUnit
Programming

Load Generator

Load Testing Tool

Virtual Users

App.

API

Network

Server

Application
Considerations

• Requires programming / access to APIs
• Tool support
  – Extensibility
  – Language support
• May require more resources
• Environment may need to be set
Real Users

• Testing in production
  – Full load
  – Partial load (A/B testing, canary testing)
• You trade in the need to generate and validate workload for a possibility of performance issues and load variability
Considerations

• May make sense for the following conditions
  – Potential issues have minimal impact on user satisfaction and company image
  – Easy rollback of the changes
  – Homogenous workload and a way to control it
  – Fully parallel and scalable architecture
Summary

• *There is no best approach* – it depends
  – More of an art in non-trivial cases

• *Does the taxonomy make sense?*
  – Any suggestions / corrections?

• *Can load generation be more of an science?*
Questions?

Alexander Podelko

alex.podelko@oracle.com
alexanderpodelko.com/blog
@apodelko