Performance Testing

An introduction.

1. Introduction
2. Performance Testing Process
3. Performance Test Types
4. Tools
5. JMeter
6. Questions
Introduction – This presentation

Introduction – Functional vs. Non functional testing

Non Functional testing
Targets aspects of the software that are not related to a specific function or user action.
E.g. performance, scalability, security, usability, maintainability, portability
Introduction – What is performance testing?

- **Performance testing** is done to validate:
  - Speed
  - Capacity
  - Scalability
  - Stability
  - Expectations
  - Environment

Introduction – Questions to answer

- Is the product ready to be released?
- Is fast enough?
- Is able to support the expected user base?
- What happens if something goes wrong?
- How can we detect or predict if something is going wrong?
- What will happen when I get more customers?
- Where is the bottleneck?
- Is better than the competition?
Why is important?

- **Google – Speed Matters**
  - 100 to 400 ms increase in latency -> 0.2% to 0.6% reduction in the number of searches/user/day (~8 million searches per day)
  - longer users exposure -> fewer searches
  - loss of searches persists even after latency returns

- **Amazon**
  - Page load slowdown of 1 second -> loss of $1.6 billion in sales/year

- **Healthcare.gov**

![Chart showing page abandonment increase as a percentage with page load time in seconds]
## Performance Testing Process

1. Understand the Project Vision and Context  
2. Learn about the Test Environment  
3. **Identify Performance Acceptance Criteria**  
4. Create the workload model*  
5. Configure the Test Environment  
6. Implement the Test*  
7. Execute the Test*  
8. Analyze Results, Report, Tune, Retest*

### Performance Testing Process – Acceptance Criteria

- **Identify or estimate**  
  - Response time (user concern)  
  - Throughput (business concern)  
  - Resource utilization (system concern)

- **Sources**  
  - Business requirements  
  - User expectations  
  - Contracts  
  - Regulations  
  - Realistic workload models  
  - Previous release  
  - Competitor’s application
Performance Testing Process

1. Understand the Project Vision and Context
2. Learn about the Test Environment
3. Identify Performance Acceptance Criteria*
4. Create the workload model*
5. Configure the Test Environment
6. Implement the Test*
7. Execute the Test*
8. Analyze Results, Report, Tune, Retest*

Performance Testing Process – Workload model

- Simulate the real-world.
- User’s behavior
  - Navigation
  - Delay patterns and distribution
  - Think time
  - User abandonment
  - Variability
  - Common user errors
- Key user scenarios
- Create user models
- Identify and Generate test data
- Identify metrics
Performance Testing Process

1. Understand the Project Vision and Context
2. Learn about the Test Environment
3. Identify Performance Acceptance Criteria* 
4. Create the workload model* 
5. Configure the Test Environment
6. Implement the Test* 
7. Execute the Test* 
8. Analyze Results, Report, Tune, Retest*

Performance Testing Process – Implement tests

- **Implement the test design**
  - Consider available application features and components
  - Start small - „one scenario at a time“
  - Pay attention to
    - Test data feed
    - Response validation
    - Response processing
    - Response time measurement
  - Match the script to the design
    - Extend selected tool or select another one if possible.
Performance Testing Process – Implement tests

- **Validate the test**
  - Check that the test does what was intended to do
  - Run each test with different user Nb. (e.g. 1, 3, 5, and 11 users)
  - Check combination of tests scenarios
  - Validate test data
  - Validate result monitoring
  - Validate result collection

Performance Testing Process

1. Understand the Project Vision and Context
2. Learn about the Test Environment
3. Identify Performance Acceptance Criteria*
4. Create the workload model*
5. Configure the Test Environment
6. Implement the Test*
7. **Execute the Test***
8. Analyze Results, Report, Tune, Retest*
### Performance Testing Process – Test execution

- Run „smoke test”
- Run tests in one- to two-day batches
  Quick feedback - Quick reaction
- Run the tests multiple times (at least 3 times.)
- Use dedicated load generation systems.
- Monitor:
  - scripts
  - systems
  - data
- Archive any information needed to reproduce the test
- Pay attention to unusual behavior (use your instincts.)

---

### Performance Testing Process

1. Understand the Project Vision and Context
2. Learn about the Test Environment
3. Identify Performance Acceptance Criteria*
4. Create the workload model*
5. Configure the Test Environment
6. Implement the Test*
7. Execute the Test*
8. Analyze Results, Report, Tune, Retest*
Performance Testing Process – Interpret and report results

• Customize reports for target audience

• Report:
  – Quickly
  – Frequently
  – Intuitively

• Try to identify trends

Performance Testing Process – Interpret and report results

• Analyze data immediately and in cross-functional team

• Double check statistics

• Make raw data available

• Use results to update the tests
  – Is the test providing enough information?
Performance Test Types

- **Baseline Testing** examines how a system performs under expected (normal) load.

- **Load Testing** verifies application behavior under normal and peak load conditions (but not breaking it).

- **Capacity Testing** finds the volume of load where the system actually breaks or is close to breaking.

-performance-Types

- **Stress Testing** test beyond normal operational capacity, often to a breaking point.

- **Spike Test** load exceeds the anticipated load for a short period of times.

- **Soak**: Long-period tests to find system instabilities (e.g. detecting memory leaks, determine MTBF, MTTF).

- **Scalability Testing** find out how performance is affected by adding additional resources (CPU, hosts).
Performance testing tools

- **Load generator tools**
  - Stand-alone load-generators
    - Apache JMeter
  - Cloud based
    - BlazeMeter
    - Loadstorm
  - Enterprise-Level solutions
    - HP Loadrunner

- **Monitoring and analysis**
  - Resource Monitors
    - Windows Perfmon
    - UNIX SAR, top/htop
    - HP Sitescope
    - VisualVM
    - Zabbix
  - Profiling
    - SQL profilers
    - CPU profilers
    - Memory profilers

JMeter

- **Primarily a load-generator tool**
  - To implement and execute performance test scenarios
  - Basic presentation of measurement data*
  - No resource monitoring or analysis capabilities*

- **Popular due to...**
  - Broad protocol coverage (HTTP, JMS, SOAP, JDBC, LDAP, FTP, POP3)
  - Open-source (extensibility, low-cost)
  - Platform-independent (written in Java)
  - Quick and visual test-composition
  - Appropriate for developers and testers too
  - Scripting "by hand" or with "traffic recording"
  - Client - Server mode
JMeter - Test Plan Structure

• **Tree structure**
  – Drag and drop elements
  – Enable, disable or save any sub tree

• **Test Plan**
  – Main Node of test
  – Overall settings
  – User defined variable
  – Specify class path settings (e.g. JDBC driver)

• **Thread Group**
  – Beginning point of any test plan
  – Defines pool of users
  – Ramp up period
  – Loop count
  – Ancestor of Controllers and Samples
### Jmeter - Test Plan – Scoping Rules

- **Ordered elements**
  - Controllers
  - Samplers

- **Hierarchical elements**
  - Listeners
  - Config elements
  - Pre/Post processors
  - Assertions
  - Timers

### Jmeter - Elements

- **Controllers** – drive the processing of the test
  - **Samplers** – send requests and wait for response (e.g. HTTP, SOAP, FTP)
  - **Logic controllers** – customize request sending (e.g. Once only, Switch, While or Loop)

- **Listeners** – provide access to information gathered by JMeter (display, Save (CSV/XML) or Reload results)

- **Timers** – add delay time between requests

- **Assertions** – check responses from the server

- **Configuration elements** – add to or modify requests

- **Pre-processor elements** – execute actions prior to a request

- **Post-processor elements** - execute actions after a request

- **WorkBench**
JMeter Performance Tuning

- Use the latest version (validate it before using)
- Use NON-GUI mode for load testing
- Fine tune JMeter Java options
- Use CSV output.
- Do not log results that are not needed.
- Use Post Processor and Assertions efficiently.
- Use Regular Expression Extractor and extract as less data as possible (do not use „Body unescaped“)
- Avoid XPath Extractor
- Use JSR 223 + Groovy for scripting + Caching
- Use external scripts if possible
- Do not overscript
- Remove all listeners during test execution
- Generate report after run

Homework

- Choose a performance test type (see slide 23-24)
- Create a JMeter test plan using HTTP recording
- Perform a maximum 30 minute long test
- Artifacts to be provided:
  - JMeter results in csv format (using Simple Data Writer)
  - 4 different charts produced using JMeter listeners
    - The following three charts shall be provided:
      - Hits per second (aggregated)
      - Response Time Over Time (aggregated)
      - Combined charts of Hit Per Second and Response Time Over Time (aggregated)
    - VisualVM screenshots of JMeter monitoring taken at the end of the test (minimum: Monitor and VisualGC)
    - Description of the test environment (Systems, initial DB content etc.)
    - Short description of what was observed

https://drive.google.com/folderview?id=0B3zy6db4TOy8OUl0dWJ2VVJ3MGc8usp=sharing
JMeter

For more information, please contact:
Zsolt Balázs
Software Test Automation Engineer
EPAM Systems, Inc.
Futó street 47-51
H-1082 Budapest, Hungary
Phone: +36 1 815-69-00 x 40078
Email: zsolt_balazs@epam.com
http://www.epam.com

Materials

- http://perftestingguide.codeplex.com/releases/view/6690
- http://jmeter.apache.org/usermanual/
- http://scott-barber.blogspot.hu/
- http://www.perftesting.co.uk/
- http://www.allthingsquality.com/p/people-in-testing.html?m=1
Appendix

• **VisualVM PC user/password: adminuser/adminuser**