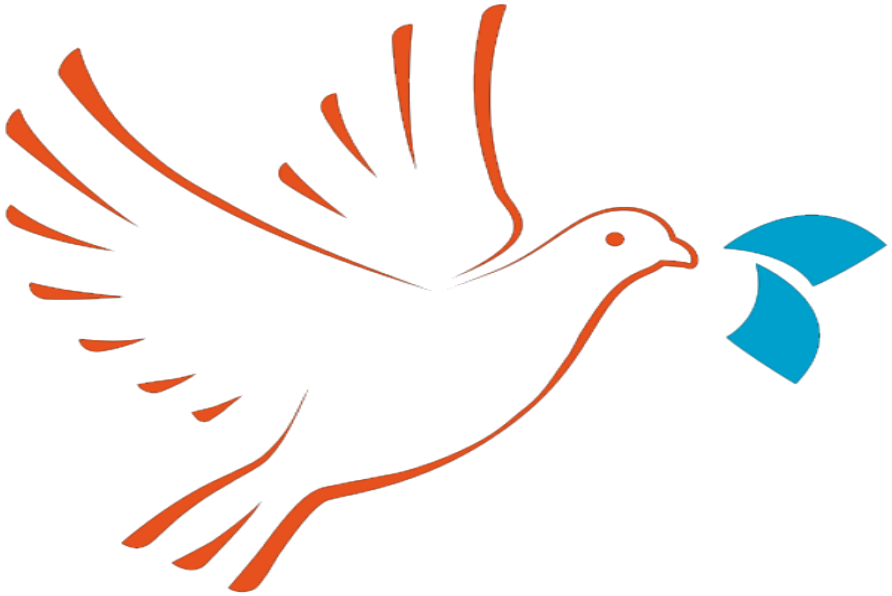

Performance Testing

Blaise

Gaining deeper understanding

Introduction

- Why do performance testing?
- What is a load test?
- How does the Blaise team execute performance tests?
- Create and run your own performance test

How many sessions can run on a Blaise Server ?

- Very general question
- Not easy to give an answer
- Many factors determine the server load:
  - Complexity of the Rules of the instrument
  - The size of an interview page (# Controls)
  - The time between two requests of a session
  - Hardware of the Server:
    - CPU
    - Memory

Solution

- Make question more specific:
- Consider a specific instrument installed on a Server (Park)
- How many sessions of this instrument can be hosted by this server park?
- Measure the actual response times

How many servers are needed?

- Maximum response time for request to server:
  - Time a respondent has to wait for the next page
  - Determine how many concurrent users can be served
- Expected maximum concurrent users
- Number of servers = max users / (max users per server)
- Needs to be determined before the instrument ‘goes live’

Load Test

- Simulate interview sessions
- Measure response times
- Validate responses
- Gradually increase number of sessions
- Goal:
  - Determine maximum concurrent sessions
  - Detect bottlenecks

How to simulate an interview session?

1. Record an interview session
2. Playback the recording

Record an Interview Session

- In Visual Studio:
  1. Create a Web Performance and Load Test Project
  2. Add a Web Performance Test:
     - Internet Explorer is started with Web Test Recorder Add-on
  3. Fill in URL of Web Survey
  4. Complete the Survey
  5. Stop Web Test Recorder:
     - The Web Performance Test is filled
- Demo

Problems with playback

- Literal playback does not work:
  - Playback session will have different session id:
    - Recorded session id cannot be used
  - Cannot use literal recording in concurrent sessions:
    - Single case modified by multiple users…
- Create initial playback script in Visual Studio:
  - Generate Code from Menu in Web Performance Test (Demo)
- Modify playback script:
  - Generate unique Primary Key Value
  - Use actual runtime Session Information

How to simulate an interview session? (2)

- Need to simulate answer time
- Do not want to use actual times of recording
- Instead use standard times:
  - Per answered Question
  - Per Page
- Modify Playback Script to use desired answer times

Modify Web Test

- Tool that automates modification of Playback Script:
  - Use runtime Session Information (obtained from Server Response)
  - Modify Key Page Request to use unique Primary Key value
  - Specify desired answer times based on number of questions on page
  - Adds Validation Rules for expected Responses
  - Modified Script is based on additional test classes
- Developed by Blaise Team:
  - Sources are available on request
- Demo

Test Playback Script

- In Visual Studio:
  1. Replace Coded Web Test by modified Web Test
  2. Build Project
  3. Specify Test Mix in Load Test (Select modified Web test)
  4. Specify Test Scenario:
     1. Constant Load Pattern (User Count: 1)
     2. Think Profile: Off (no waiting time between requests)
  5. Specify Run Settings:
     1. (Use) Test Iterations: 1
  6. Run Load Test

Test Playback Script (2)

- Inspect results of single user run:
  - No errors or Failed Tests ?
  - Has Record been added to Blaise Database?

Run Playback Script

- In Visual Studio:
  1. Specify Test Scenario:
     1. Step Load Pattern (Maximum User Count: 200)
     2. Think Profile: Normal Distribution
  2. Specify Run Settings:
     1. Use Test Iterations: False
     2. Run duration: 01:00:00
  3. Specify initial Primary Key Value
  4. Run Load Test

How to interpret result of Load Test?

- Open Load Test Report (Summary)
- Overall Results:
  - Tests Failed: 0 ?
  - Avg. Response Time
- Page Results:
  - Avg. Page Time (execute action)
  - Compare with desired Maximum Time
  - Change Max User Count for next Load Test, if needed
- Repeat Load Test to estimate Maximum User Count

File Edit View Project Build Debug Team Test Load Test Analyze Tools Extensions Window Help Search (Ctrl+Q) WebAndLoadTestProject1

Debug Any CPU Start Live Share ADMIN

LoadTest1.loadtest | LoadTest1 [16:47] | SessionInfo.cs | WebTest4Coded_m.cs | Source Control Explorer | WebTest3Coded_m.cs | WebTest4.webtest

Summary Graphs Tables Detail

Test Completed

Load Test Summary

Test Run Information

Load test name	LoadTest1
Description	
Start time	26/08/2020 16:47:58
End time	26/08/2020 16:52:58
Warm-up duration	00:00:00
Duration	00:05:00
Controller	Local run
Number of agents	1
Run settings used	Run Settings1

Key Statistic: Top 5 Slowest Pages

URL (Link to More Details)	95% Page Time (sec)
http://localhost/flight59/	0,059
http://localhost/flight59/api/applicatio...	0,057
http://localhost/flight59/api/applicatio...	0,055
http://localhost/flight59/assets/runtime...	0,0010

Key Statistic: Top 5 Slowest Tests

Name	95% Test Time (sec)
WebTest3Coded	124

Overall Results

Max User Load	50
Tests/Sec	0,33
Tests Failed	0
Avg. Test Time (sec)	109
Transactions/Sec	0
Avg. Transaction Time (sec)	0
Pages/Sec	3,75
Avg. Page Time (sec)	0,031
Requests/Sec	7,75
Requests Failed	0
Requests Cached Percentage	25,4
Avg. Response Time (sec)	0,016
Avg. Content Length (bytes)	57.515

Test Results

Name	Scenario	Total Tests	Failed Tests (% of total)	Avg. Test Time (sec)
WebTest3Coded	Scenario1	100	0 (0)	109

Page Results

URL (Link to More Details)	Scenario	Test	Avg. Page Time (sec)	Count
http://localhost/flight59/api/application/executeaction	Scenario1	WebTest3Coded	0,042	675
http://localhost/flight59/api/application/start_interview	Scenario1	WebTest3Coded	0,028	150
http://localhost/flight59/	Scenario1	WebTest3Coded	0,015	150
http://localhost/flight59/assets/runtimeParameters.json	Scenario1	WebTest3Coded	0,00021	150

Transaction Results

Name	Scenario	Test	Response Time (sec)	Elapsed Time (sec)	Count

System Under Test Resources

Machine Name	% Processor Time	Available Memory at Test Completion (Mb)

Solution Explorer

Search Solution Explorer (Ctrl+;)

- Solution 'WebAndLoadTestProject1' (1 of 1 project)
  - Solution Items
    - Local.testsettings
  - WebAndLoadTestProject1
    - Properties
      - AssemblyInfo.cs
    - References
    - CustomPageValidationRule.cs
    - LoadTest1.loadtest
    - SessionInfo.cs
    - WebTest1.webtest
    - WebTest1Coded_m.cs
    - WebTest2.webtest
    - WebTest2Coded_m.cs
    - WebTest3.webtest
    - WebTest3Coded_m.cs
    - WebTest4.webtest
    - WebTest4Coded_m.cs

Solution Explorer | Team Explorer

Properties

Output

Show output from: Portability Analysis

Web Publish Activity | Error List ... | Test Results | Output

Ready

Add to Source Control

Performance testing for Blaise Releases

- Validate performance of (Server) Runtime of new major Blaise Release:
  - When performance dropped:
    - Determine cause (change in code)
    - Improve implementation
    - Redo load test
- Standardized Load test:
  - Fixed Survey (GEZO)
  - Fixed playback script with standardized answer times
  - Session duration: 30 minutes
  - Servers installed with Blaise Release to test

More advanced testing

- Combine multiple Playback scripts:
  - Open Load Test In Visual Studio:
  - Edit Test Mix: specify playback scripts
  - Demo
- Use preloaded data:
  - Specify Case information in file
  - Single Instance provides access to Case Information
  - Playback script gets Case Information at start:
  - Primary Key Value and Answers to some Fields are based on Case Information

More advanced testing (2)

- Run test on Multiple Web Servers
  - Make a list of (names of) web servers
  - Define variable ServerName in Playback Script:
    - Replace recorded server name (in URLs) by ServerName
    - Fill ServerName at Start of Playback Script (using list)
  - Demo

Do it yourself

- Requirements:
  - Visual Studio Enterprise:
    - Microsoft stops supporting Web Performance and Load Test Projects
    - Visual studio 2019: last version with Web Performance and Load Test Projects
  - ModifyWebTest Tool:
    - Obtain sources from Blaise Team

Alternative Load Test Tools

- Load test tools for .Net:
  - Neoload,
  - Micro Focus Silk Performer,
  - Micro Focus Load Runner

- Open source tools that support code-based tests:
  - Artillery,
  - Gatling,
  - K6,
  - Locust

Questions?