Web Application Stress Testing with Blaise Internet

Jim O’Reilly
Westat

International Blaise Users Conference
Annapolis Maryland, USA
September 28, 2007
Introduction

- **Focus:** web application stress testing tools, methods, tests and lessons learned
- **Benefits of stress testing**
  - Realistic simulation of performance of server and application under varying numbers of concurrent users
  - Valuable information on suitability of the server configuration for specific requirements of a survey
  - Potential in usability testing and paradata analysis
- **Test results presented here highly dependent on**
  - Size and complexity of the data models
  - Server platform
  - Your mileage may vary
Planning and Configuring a Web Survey

- Calculating server capacity essential
  - Respondent cooperation crucial
  - Web surveys often long and complex
  - Lags processing pages on server drive Rs away
- Server capacity complex
  - Processors, memory, speed etc.
- Application performance that meets needs of study is core
  - Done by stress testing
- Blaise 4.8 internet system enhancements offer improved scalability -- widening potential uses of Blaise
Peak Concurrent Users

- Key metric for server configuration
  
  Klaus Salchner: http://aspnet.4guysfromrolla.com/articles/062304-1.aspx#postadlink

\[
\text{Users per day} = \frac{1,000 \text{ users}}{14 \text{ hours}} = 72 \text{ users/hour}
\]

\[
\text{Supported users per hour} = \frac{4 \text{ users/hour}}{15 \text{ min}} = \frac{60 \text{ min}}{60 \text{ min}} = 4 \text{ users/hour}
\]

\[
\text{Concurrent users} = \frac{72 \text{ users/hour}}{4 \text{ users/hour}} = 18
\]

\[
\text{User per day} = \frac{hours \text{ per day} \times 60 \text{ min}}{session \text{ length in min}} \times \text{concurrent users} = \frac{14 \times 60 \text{ min}}{15 \text{ min}} \times 18 = 1008 \text{ users}
\]

- Peak = three times average concurrent users
Microsoft Web Application Stress Tool

- Freeware, unsupported by Microsoft
  - Google “web application stress testing” for links
- Capable, adaptable, suits Blaise testing situations
How WAS works

- Runs on user workstation using MS Access database
  - Records web application session
    - Captures all http traffic
    - Messages generated by user and responses returned
  - Testing a script
    - Set parameters--# of users; test times, etc.
    - As script runs, WAS records timing and result codes of script-server interactions
  - Reports on tests generated
  - One workstation can simulate hundreds of web users
Stress Test Methodology

- Follow Blaise team approach reported in Blaise 4.8 Online Assistant at “Test results for web server performance”
- Key measure: increase in waiting time per page between
  - Baseline WAS test with one user and
  - Tests of incrementing levels of concurrent users
Test Steps

- Record Blaise Internet survey session script with WAS
- In WAS
  - Set script’s page delay to a constant value reflecting expected average time users take reading, thinking, entering, and pressing Next
  - 10 seconds per field on a page used (for pages that post to the IS page handler)
- Run
  - benchmark single user test for one hour
  - successive one hour tests increasing # of users
- Extract count of pages sent from reports
- Calculate results for session series
## Test series

### Table 1: Data models & WAS scripts

<table>
<thead>
<tr>
<th></th>
<th>Fields</th>
<th>Signals/Checks</th>
<th>BMI size (kb)</th>
<th>IS Pages</th>
<th>Script Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRFS</td>
<td>41</td>
<td>0</td>
<td>18</td>
<td>20</td>
<td>17</td>
</tr>
<tr>
<td>Wes1</td>
<td>640</td>
<td>29</td>
<td>543</td>
<td>83</td>
<td>31</td>
</tr>
<tr>
<td>Wes2</td>
<td>370</td>
<td>22</td>
<td>297</td>
<td>221</td>
<td>177</td>
</tr>
</tbody>
</table>

Server: one Intel Xeon 1.3 Ghz processor and 1Gb of RAM, running Windows 2003 Server
Table 2: Stress test of BRFS
(60 minutes with average posted page delay of 22.2 seconds)

<table>
<thead>
<tr>
<th>Users</th>
<th>Pages</th>
<th>TPP*</th>
<th>Extra TPP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>536</td>
<td>6.72</td>
<td>0.0</td>
</tr>
<tr>
<td>25</td>
<td>13268</td>
<td>6.78</td>
<td>0.07</td>
</tr>
<tr>
<td>50</td>
<td>25016</td>
<td>7.20</td>
<td>0.48</td>
</tr>
<tr>
<td>60</td>
<td>28140</td>
<td>7.68</td>
<td>0.96</td>
</tr>
<tr>
<td>70</td>
<td>30580</td>
<td>8.24</td>
<td>1.52</td>
</tr>
<tr>
<td>80</td>
<td>30568</td>
<td>9.42</td>
<td>2.71</td>
</tr>
<tr>
<td>90</td>
<td>30876</td>
<td>10.49</td>
<td>3.78</td>
</tr>
<tr>
<td>100</td>
<td>25397</td>
<td>14.17</td>
<td>7.46</td>
</tr>
<tr>
<td>200</td>
<td>25555</td>
<td>28.17</td>
<td>21.46</td>
</tr>
<tr>
<td>250</td>
<td>27620</td>
<td>32.59</td>
<td>25.87</td>
</tr>
</tbody>
</table>

*TPP = (Min*60*Users)/Pages
Table 3. Stress Test of Wes1 and Wes2 (60 minutes, average posted page delay of 23.2 & 11.8 seconds)

<table>
<thead>
<tr>
<th>Users</th>
<th>Pages</th>
<th>TPP*</th>
<th>Extra TPP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>454</td>
<td>7.93</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>2256</td>
<td>7.98</td>
<td>0.0</td>
</tr>
<tr>
<td>10</td>
<td>4449</td>
<td>8.09</td>
<td>0.2</td>
</tr>
<tr>
<td>15</td>
<td>6368</td>
<td>8.48</td>
<td>0.6</td>
</tr>
<tr>
<td>20</td>
<td>7659</td>
<td>9.40</td>
<td>1.5</td>
</tr>
<tr>
<td>25</td>
<td>8513</td>
<td>10.57</td>
<td>2.6</td>
</tr>
<tr>
<td>30</td>
<td>8845</td>
<td>12.21</td>
<td>4.3</td>
</tr>
<tr>
<td>35</td>
<td>8882</td>
<td>14.19</td>
<td>6.3</td>
</tr>
<tr>
<td>40</td>
<td>8922</td>
<td>16.14</td>
<td>8.2</td>
</tr>
<tr>
<td>45</td>
<td>9184</td>
<td>17.64</td>
<td>9.7</td>
</tr>
<tr>
<td>50</td>
<td>9336</td>
<td>19.28</td>
<td>11.4</td>
</tr>
</tbody>
</table>

\[*TPP = (\text{Min} \times 60 \times \text{Users}) / \text{Pages}\]
Wes1 & Wes2 Test Comparison

Increased Seconds Per Page by Number of Users
Wes1 and Wes2 Applications

Increased TPP

Users/Threads

Wes1
Wes2
Wes1 & Wes2 Test Comparison (2)

- Better Wes2 performance
  - 40% fewer fields, less complex rules, more pages
- Follow up plans
  - Add second VM server running as IS rules server
  - More powerful server configurations

<table>
<thead>
<tr>
<th></th>
<th>Fields</th>
<th>Signals/Checks</th>
<th>BMI size (kb)</th>
<th>IS Pages</th>
<th>Script Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRFS</td>
<td>41</td>
<td>0</td>
<td>18</td>
<td>20</td>
<td>17</td>
</tr>
<tr>
<td>Wes1</td>
<td>640</td>
<td>29</td>
<td>543</td>
<td>83</td>
<td>31</td>
</tr>
<tr>
<td>Wes2</td>
<td>370</td>
<td>22</td>
<td>297</td>
<td>221</td>
<td>177</td>
</tr>
</tbody>
</table>
Blaise Team’s Tests

- Reported in OLA
- Datamodel of 1900 fields with routing and checks
- Peaked at
  - 50 users with no dedicated rules server
  - 100 users with one rules server
  - 150 users with two rules server
- Server with two 2.8 GHz processors, 4 GB RAM
WAS and Other Types of Application Testing

- Question raised on how an application would feel to a user if run at the same time as a WAS test.
- Ran Wes2 with 15 users and launched another survey interview interactively. Impressions:
  - Performance OK -- page delay ~1 second, occasionally 3-4 seconds
  - Found functional anomaly; in enumerated items, up/down arrow key in browser didn’t work
    - SN unable to replicate the problem
  - Suggest possible use in usability testing?
  - Check app look and feel under stress
WAS and COM

- WAS has COM interface; open to automation
- Built simple VB app to run series of tests

- Enables high volume testing
WAS Database

- Detailed data on tests
- MS Access 95
  - Difficult to work with directly
- WAS provides access to script pages to change ResponseDelay etc.
- VB/other apps can access directly
- Mdb also flexible for reading or export data using SQL Server and Crystal Reports etc.
Other Metrics for Blaise Internet Stress Testing

- Blaise team’s stress testing approach
  - WAS scripts, fixed page delays, one hour tests, examining time per page at different user levels
  - Elegant, easy to implement and track, especially with WAS automation
- Tracking interviewing information during WAS series
  - Other possible insights on process
  - # of completed cases
  - Stats on interview times – avg, min, max, stddev
  - Journal data on for every page sent from server
  - Page-level stats of potential value – timing distribution, outliers etc – as indicators of page structure issues
Linking interviewing and WAS testing

- In workshop enable journaling process
- Set data model primary key to tie interview to journal
- Modify interview starter page to assign web server session id as primary key

```vbnet
Function CreateOptions(Hosts)
    '
    If UseRunTimeOptions Then
    '
        '*** setting sessionid as primary key
        Dim sCaseID
        sCaseID = Session.SessionID
        RootNode.setAttribute "KeyValue", sCaseID
        '***
    '
End Function
```
Conclusions

- Stress testing Blaise Internet with WAS system
  - Relatively easy to implement and use
  - Valuable approach to understanding the expected performance of Blaise Internet for both specific applications and different server configurations.
- Able to demonstrate to clients whether the Blaise web survey system is ready to meet their needs.
- Potential for
  - usability testing
  - processing or paradata analysis with journal information