Web Application Stress Testing with Blaise IS

Jim O'Reilly, Westat

1. Introduction

Survey data collection over the Internet is an increasingly important technique. With the release of Blaise 4.8 in July 2007, the Blaise community has a much improved and more powerful system for web surveys. Blaise 4.8 enables development of web questionnaires and administration of surveys across a wide range of requirements—from small to large sized questionnaires and from small to large sized numbers of survey takers.

Central to web survey implementation is providing the web server infrastructure to support peak numbers of survey users. With version 4.8, Blaise has made major improvements to server-side responsiveness, robustness and scalability. An individual web server can support all functions for small to medium applications and user volume. For large scale situations multiple servers can be integrated to support web, database, and rules services suitable for the application.

Determining in advance whether a given server system is sufficient to support a web survey's requirements is critical for planning and configuration. Web stress testing systems are designed to address this important issue. We discuss here methods and applications using a web application stress test system with Blaise IS.

While we present results of stress tests, our main focus is the methods used. The results are highly dependent on the size and complexity of the data models as well as the server platform. In other words, your mileage may vary.

2. Microsoft Web Application Stress Tool

We used the Microsoft Web Application Stress tool (WAS). This is a freeware product distributed by Microsoft. While a legacy product that has not been updated in years, and which Microsoft offers without any support, WAS is a reasonably capable tool that can be adapted to a number of applications. WAS seems to suit Blaise IS testing situations. To obtain WAS, and for instructions on its use, a Google search of "web application stress testing" returns a variety of useful links.

2.1 Using WAS

Here is an overview of the WAS testing process:

- WAS runs on a user workstation and records a web session. All the http traffic of the session—messages generated by the user and sent to the server and the responses returned by the server—are written to the WAS database. The recorded series is called a script.
- To test using a script, one sets parameters—number of users (called threads), test time, and others. When the test is started in WAS, simultaneous web sessions to the server are generated, sending for each the script's recorded messages in sequence after a preset and/or random delay. The time and result of the return

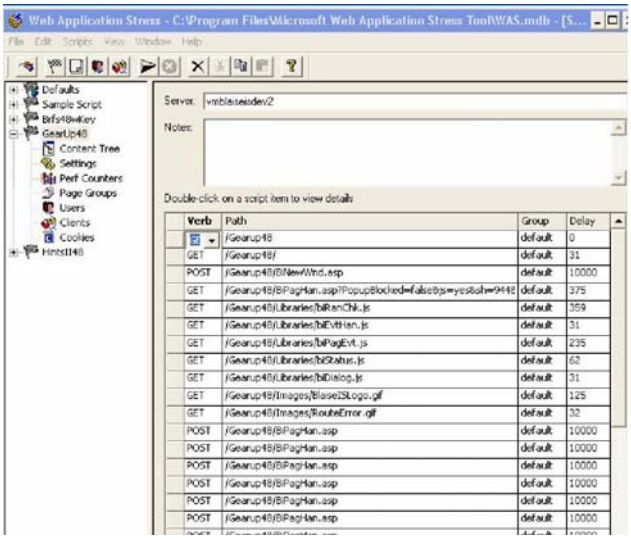
message from server is saved. As a thread is completed for a session, a new one is started until the specified test time is reached.

- Following the test, WAS produces a series of reports on the web interactions--result codes and counts, time for the requested page to return to the workstation, and more.
- A single workstation can simulate tens or hundreds of web survey users for lengthy periods and provide valuable data on server performance.

3. Blaise IS Stress Test Model

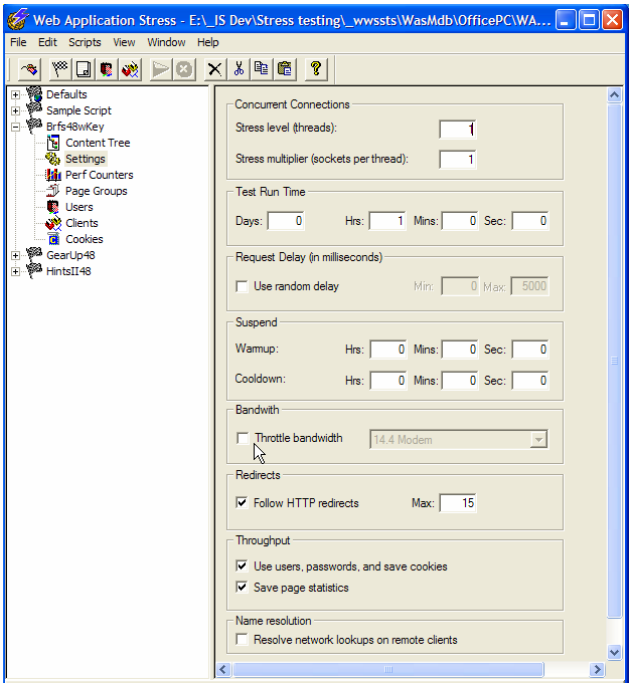
Our testing follows the method used by the Blaise team for web server performance testing reported in the Blaise 4.8 Online Assistant. The key metric is the additional waiting time per page as the number of concurrent users increases. The steps for testing a web application are:

- Record the Blaise IS survey session in WAS.
- In the WAS application, set the script's page delay value to 10 seconds per field on a page for pages that are posts to the biPagHan.asp—the IS page handler.

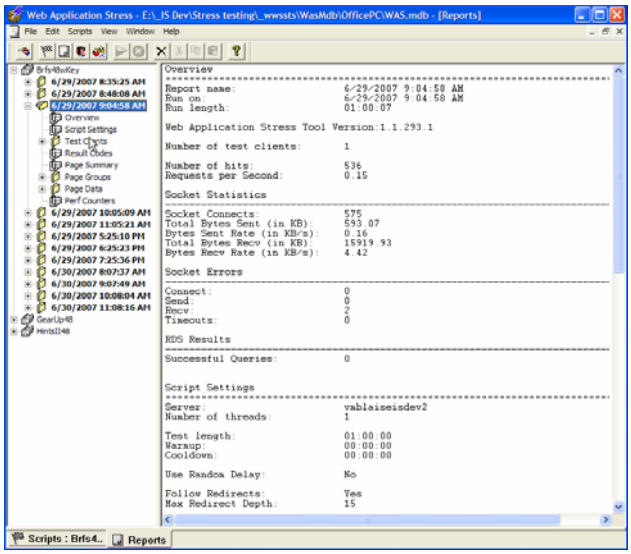


A page with one response field would wait ten seconds before WAS transmits the recorded response to the server. The ten seconds delay per response field is the assumed time required for the average user to read the item, decide on a response, enter the response, and transmit it to the server.

- With the script ready, one sets the parameters for a single-user benchmark test
    - 1 thread (user)
    - 1 hour test time



- After the test is run the report is examined checking that there are few connect errors and recording the socket connects, which is the number of pages received from the server by the browser.



The test time divided by the page count gives the time per page for the baseline single-user situation. For the one hour test of one thread this is 3600 / Pages.

- Re-run the script successively, increasing the number of threads and record the page count and compute the time per page per user (3600 * Threads) / Pages

3.1 Tests

We conducted a series of tests using three different data models

Table 1: Data models and WAS Scripts

	Fields	Signals/ Checks	BMI size (kb)	IS Pages	Script Pages	Script Pages	Page Delay (s)
Brfs48wKey	41	0	18	20	17	17	22.2
Wes1	640	29	543	83	31	31	23.2
Wes2	370	22	297	221	177	177	11.8

BRFS uses the core items of the well known Behavioral Risk Factors Survey instrument (www.cdc.gov/brfss) of the U.S. Centers for Disease Control and Prevention. Wes1 and Wes2 are web surveys that were conducted by Westat using Blaise IS 4.7.

The server used for these tests had one Intel Xeon 1.3 Ghz processor and 1Gb of RAM, running Windows 2003 Server. WAS ran from a desktop workstation. In the tests reported below page delay is the average number of data fields on the interview pages times ten seconds. This is a substitute for changing the page delay in the script page by page to reflect the fields on the page. In the Wes1 and Wes2 datamodels large number of pages made it difficult to identify exactly what page in the script corresponds with a page in the IS application.

Table 2: Stress test of BRFS (60 minutes with an average page delay of 22.2 seconds)

Users	Pages	TPP*	Extra TPP
1	267	13.48	
25	6493	13.86	0.38
50	12687	14.19	0.70
100	24507	14.69	1.21
125	26565	16.94	3.46
150	29255	18.46	4.98
175	30284	20.80	7.32
200	31503	22.85	9.37
250	30429	29.58	16.09

*Time per page (TPP): (Min * 60 * Users) / Pages

As the number of users increase the performance declines, of course. But at what level of responsiveness (extra TPP) does it become a problem? Two seconds is used as the threshold. So for this survey with all functions (web, database, rules) running on one server of modest capacity (processors and RAM), the peak number of users that can be handled with an average time per page of less than two seconds greater that the single-user baseline is about 100 (the shaded cell).

For the much larger Wes1 data model, the threshold of two seconds of average extra time per page is reached between 40 and 45 users (Table 3). And for the second large data model, Wes2, the threshold is between 35 and 40 users (Table 4).

Table 3. Stress Test of Wes1 (60 minutes with an average page delay of 23.2 seconds)

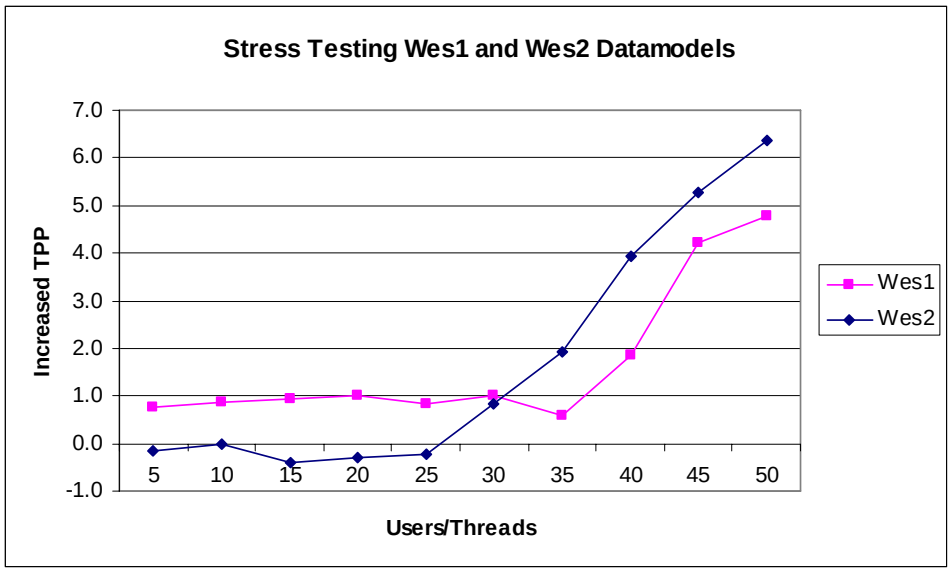
Users	Pages	TPP*	Extra TPP
1	203	17.73	
5	973	18.50	0.8
10	1936	18.60	0.9
15	2892	18.67	0.9
20	3842	18.74	1.0
25	4852	18.55	0.8
30	5764	18.74	1.0
35	6883	18.31	0.6
40	7350	19.59	1.9
45	7386	21.93	4.2
50	7990	22.53	4.8

*Time per page (TPP): (Min * 60 * Users) / Pages

Table 4. Stress Test of Wes2 (60 minutes with an average page delay of 11.8 seconds)

Users	Pages	TPP*	Extra TPP
1	356	10.11	
5	1808	9.96	-0.2
10	3570	10.08	0.0
15	5553	9.72	-0.4
20	7323	9.83	-0.3
25	9087	9.90	-0.2
30	9879	10.93	0.8
35	10482	12.02	1.9
40	10263	14.03	3.9
45	10536	15.38	5.3
50	10918	16.49	6.4

*Time per page (TPP): (Min * 60 * Users) / Pages



In examining the different test experiences of the Wes1 and Wes2 datamodels, the better performance of Wes2 may be explained by its lesser complexity—about 40% fewer fields and significantly less complex rules.

Follow-up activities are underway to enhance the server configuration, re-run the tests and examine the impact. A second IS server dedicated to running the rules will be added. The possibility is also being explored of using a more powerful server with more processors, speed, and RAM.

The Blaise team in the Blaise 4.8 Online Assistant reports in the section "Test results for web server performance" the significant impact of adding one or more dedicated rules servers. They used a large 1900-field datamodel with routing and checks. It peaked (< 2 seconds extra TPP) at 50 users with no dedicated rules server. Adding one rules server increased the peak to 100 users and adding a second increased the peak to 150 users.

3.2 An Ad Hoc Test

A colleague in looking at the test results questioned what an IS interview would look and feel like to a person when a WAS session was active. We tested this. Running the Wes2 script for 15 users, where the average extra TPP is .6 seconds, we independently conducted an interview on the application.

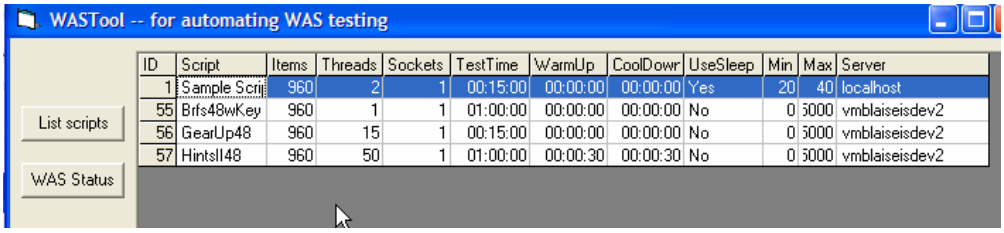
The performance was acceptable with most pages being returned by the server in a second or so. Occasionally the lag was greater, 3 or 4 seconds. An occasional delay of this duration felt normal by everyday web experience. Also, these delays seemed not inconsistent with the average delay time reported by WAS.

In this test we did encounter one functional anomaly. In all radio-button group items, the down-arrow key would not move the cursor in the list. One had to use the mouse to select. We reported this to the Blaise team.

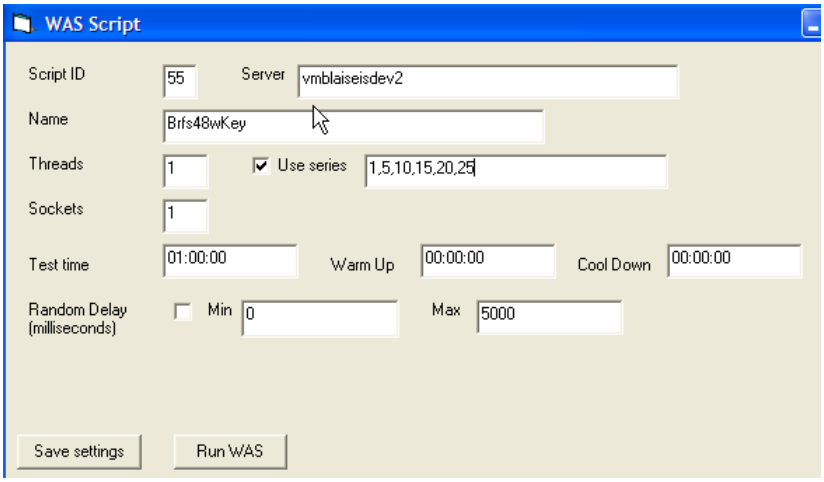
This finding suggests that WAS might be valuable for usability testing to insure the application performs correctly when a number of other users are also active.

3.3 WAS and COM

A useful feature of WAS is that it has a COM interface, so one can automate it. We developed a VB application called WASTool to support running a series of tests. The main form displays the recorded scripts in the WAS database.



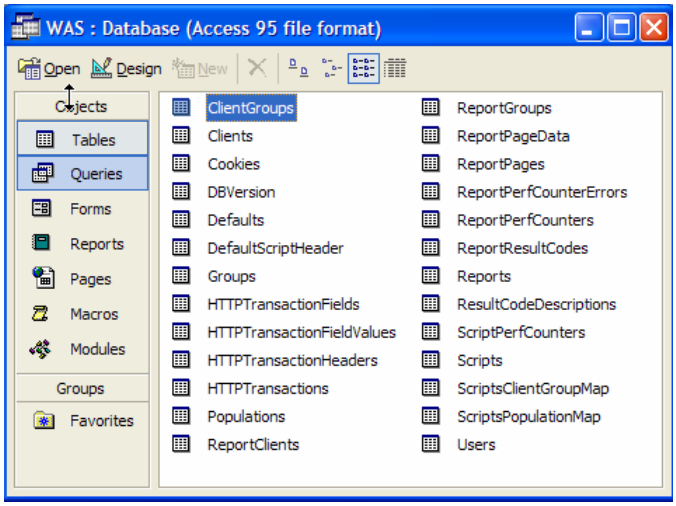
On clicking on a script, a form presents editable test parameters of the script—server, threads, test time, etc. The 'Use series' check box .and the edit box next to it are added automation parameters.



If Use series is checked and the edit box has a comma-separated array of thread values, when Run WAS is pressed, WAS runs the script for each thread value in the series.

3.4 WAS Database

WAS uses a MS Access 95 database. The names of the tables in WAS.mdb suggest the breadth and detail that is recorded.



Other than changing a few script parameters in WASTool to support automated tests, we have only read or exported data from WAS.mdb using other systems such as SQL Server and Crystal Reports for reporting and other activities.

4. Other Metrics for IS Stress Testing

The Blaise team's stress testing method-- using WAS scripts with fixed page delays (10 seconds per response field on the page), one hour tests, and examining the time per page at different user levels-- is elegant and comparatively easy to implement.

We have done some work looking at Blaise interview records produced during WAS interview sessions as a supplement. After all, it is the interview data that we are ultimately concerned with. While we are not at a stage to present these statistics, it might be useful to discuss some of the techniques and issues.

To be able to identify test interview cases, it is important to have a case id primary key for the datamodel. However, WAS does not allow passing parameters to scripts. It runs the recorded script as is. Fortunately, in Blaise IS it is possible to adapt the interview starter page, biInterviewStarter.asp, so that when an interview is launched a unique server-generated session ID is set as the primary key for the interview case. The code is shown below.

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

Also, the IS journaling function must be implemented. With a journal enabled, a separate journal database is used and the IS server produces audit trail-type records during the interview process. When the interview starts and ends and when each page is sent to the user the journal records the date and time, primary key, server session id, and information on the action, mainly the page number and number of fields on the page.

The journal information allows one to compute interview times for cases in the IS application data file without having to add code to the data model. With this one can expand the information for each WAS test, adding to the average page time per thread level the number of completed Blaise interview and the statistics on the interview time: average, minimum, maximum, standard deviation.

As well, it may be possible to use the detailed page timings to learn other things about the process. This resembles the Windows Blaise audit trails, but it is much more limited because we know nothing about what the user did on any page, only when each page was sent from the server.

It seems possible, however, that analyzing the timing statistics by page for all cases in a study might by useful in identifying pages with different distributions of timing or outlier cases on key pages.

The steps being taken to test this approach are:

- Copy the application data and journal data files from the IS server.
- In Manipula export to ascii, appending fields on the application name and processing group.
- Import the ascii files into Sql Server tables
- Update the application and journal data records with the WAS.mdb report ID for the test based on the data time stamps and the WAS reports test start and end time

With interview data, journal, and WAS report results integrated it becomes possible to explore a variety of questions about the IS system, the application, and groups of interviews.

5. Conclusions

Stress testing of Blaise IS using the WAS system is a valuable approach to understanding the expected performance of Blaise IS for both specific applications and different server configurations. With major improvements in performance and scalability of Blaise IS in version 4.8, this testing process can demonstrate to external and internal clients whether the Blaise web survey system is ready to meet their needs. In addition to these top level issues, the testing process seems applicable to usability testing and using the journal information for process or paradata analysis.