Performance and security enhancements on the Blaise IS standard stylesheet

Arnaud Wijnant, Edwin de Vet – CentERdata, Tilburg University, The Netherlands

1 Abstract

CentERdata has several years of experience in administering Blaise surveys over the Internet. In 2010 we migrated to Blaise IS, the standard CAWI server included in the Blaise installation.
The Blaise IS package consists of 3 parts that cooperate to fulfill the task of hosting a web questionnaire. These parts are the web server which creates the pages for the respondents, the data server that manages the answers given by the respondents and the rules server that applies the rules defined in the Blaise questionnaire. They can be replicated on several servers to increase the capacity of a questionnaire.
For some surveys that contain large tables and groupings of questions we found that the web server component required a high amount of system resources per respondent. This was mainly caused by the transformation of the XML output of the questionnaire engine to the HTML output of the web server. This transformation is done via a XSLT style sheet that is shipped in the Blaise installation. We customized this style sheet to obtain the look and feel we wanted, but the style sheet contained a lot of functionality and overhead that we do not need. To resolve this issue, we developed a smaller and simpler style sheet from scratch. This style sheet contains only the needed functionality, is easier to maintain and produces simpler HTML output. Load tests show that we can serve more respondents per server with our self created style sheet.
We also implemented extra security features in Blaise IS. In our setup, respondents provide the primary key as a KeyValue parameter (either GET or POST) at the start of the interview. We developed a simple mechanism to verify the identity of the respondent using a hash value based on this KeyValue.

2 Introduction

At our institute we run web questionnaires using Blaise IS for internet panels consisting of several thousands of people. At peak times, approximately 10 respondents per minute start a questionnaire and answer 5 questions per second on average. During several of these peak times, we noticed a strong drop in performance: respondents sometimes had to wait for over 10 seconds to get a new page or could not access the server at all. This drop in performance was especially problematic with a particular questionnaire that contained lot of tables.

3 Current situation

To be able to solve this problem, we first had to identify the cause of the problem. For this we first looked into our server park. Our Blaise IS server park consists of 2 combined web and rule servers and one central data server. During peak hours we could identify that the problem was not on the data server, since the CPU load on this server was negligible.

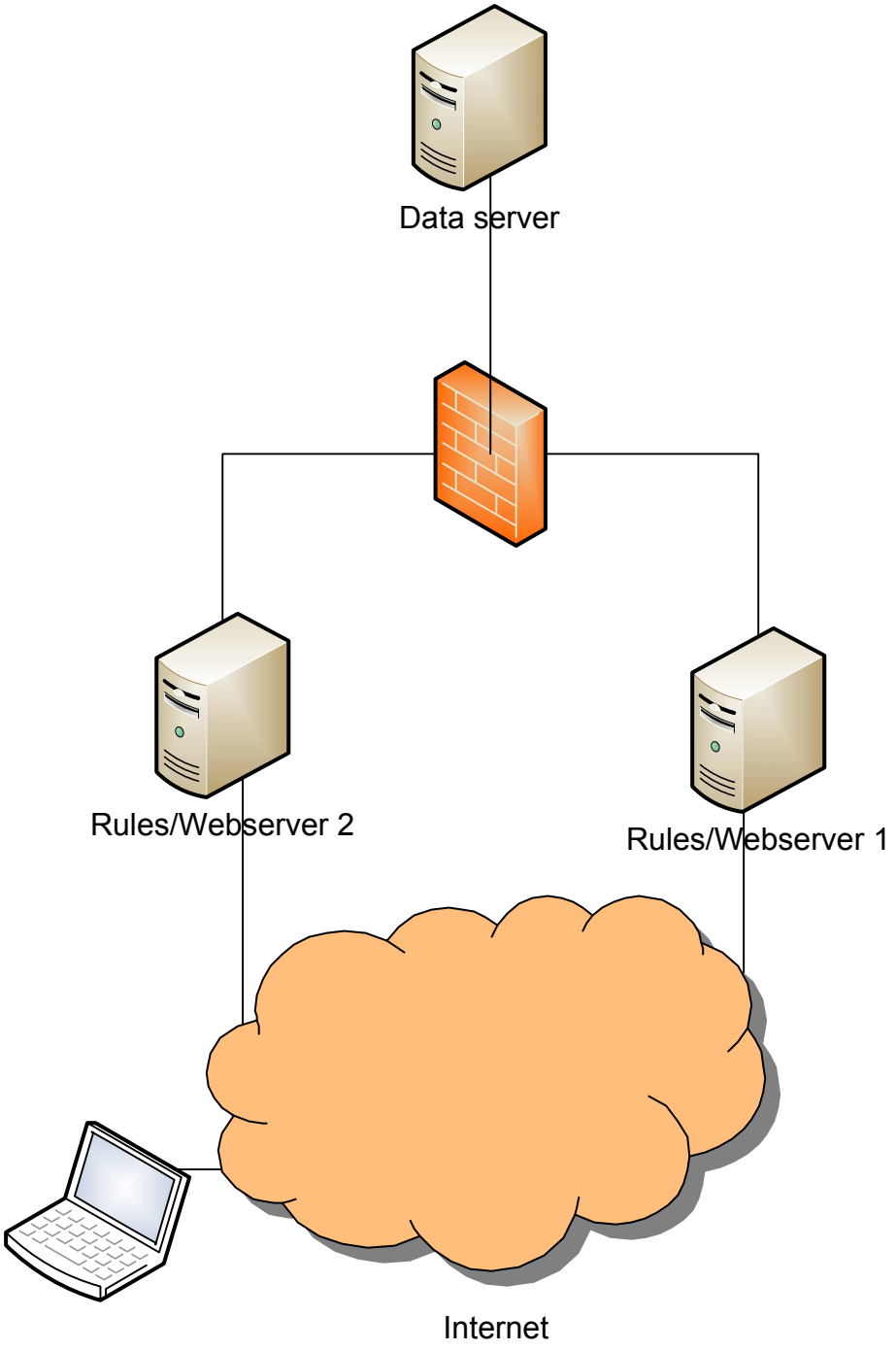

Figure 1 The Blaise IS server park

The problem had to be on the combined data and web-services. In our setup we balance the load between the webservers by assigning questionnaires to a specific web server. This means that every questionnaire is only visited through one of our 2 web servers. Since the performance problems were only on one of our servers, it was likely that the problems were questionnaire specific. It was also clear that the problems were related to the web server and not the rules server, because there was only one process that had a heavy load on the system and this was the process of the web server.
The next step in the process was to find out which of the questionnaires was causing the problem. We observed this by isolating the questionnaires on a separate server. The high CPU load was only observed on a questionnaire with pages that contain large tables of radio buttons. We therefore suspected that the XSL transformation was the bottleneck in the generation of the output. The Blaise IS XSL style sheet included in Blaise is rather big (around 6000 lines) and the output is in our opinion too large and complex. We often observe highly nested tables and div elements.

4 Solution

In order to investigate whether we could improve the performance and simplify our HTML output, we wrote a simplified style sheet from scratch with only the required functionality. This simple style sheet is roughly 6 times smaller than the original style sheet. For this reason the style sheet is easier to read and modify and the HTML output is much simpler (hardly any use of nested html tables and div elements). This simplicity is accomplished by removing unneeded functionality. However there are also new features present, for instance, we use the label tag for text belonging to checkboxes and radio buttons, which makes the text clickable in most modern browsers. Also, the generated question pages do not rely on JavaScript. The settings in the .bml and .bmf files are often ignored. Changes to the Look and Feel should be done directly in the style sheet.

5 Analysis of solution

To test our style sheet in a systematic way, we used a tool developed by MicroSoft and called Web Capacity Analysis Tool (WCAT). ”Web Capacity Analysis Tool is a lightweight HTTP load generation tool primarily designed to measure the performance of a web server within a controlled environment. WCAT can simulate thousands of concurrent users making requests to a single web site or multiple web sites. The WCAT engine uses a simple script to define the set of HTTP requests to be played back to the web server. Extensibility is provided through plug-in DLLs and a standard, simple API.” [1]

For our setup we wrote a script which perfectly mimics a respondent that fills in the questionnaire with which we had the performance problems. The simulated respondent answers a page with questions every 5 seconds. The script is loaded into the WCAT system and simulates the visit of several respondents. For our test we did test with 150 and 200 simultaneous respondents.

The server that was used to host the questionnaire for this load test is the following:

Processor: Quad-Core AMD Opteron(tm) processor 2369 HE 2.40 GHz

Installed memory: 4.00 GB

Operating system: Windows Server 2008 R2

No other questionnaires were in use at this server during the load test.

5.1 Results

The performance can be measured in several performance parameters. For our load test we present two of them: the number of requests/second being handled by the server and response times.

The number of requests per second is the number of pages that were generated for respondents during the load test. For a test with 150 simultaneous respondents that ask for a page every 5 seconds the ideal number for this parameter will be 150/5 = 30 requests/second. This number is only achieved if the generation of the page is done in 0 seconds. For the test with 200 simultaneous respondents the ideal number is 200/5 = 40 requests/second.

The table below shows the outcomes for the load tests that we did with both the original and simple style sheet.

	Standard style sheet	Simple style sheet
150 sim. respondents	26.12 req./sec.	28.81 req./sec.
200 sim. respondents	5.51 req./sec.	33.88 req./sec.

For 150 simultaneous respondents we see that the simple style sheet handles a bit more requests per second. Both numbers are close to the optimal number of 30 requests per second, so no major problems occur. However for the 200 simultaneous respondents we see a huge difference in the number of handled requests per second. Because the server with the standard style sheet is overloaded with requests, the time it takes to process one requests increased significantly.

Another parameter to look at is the response time of the server. The following table shows the maximum response time and the average response time of every request.

Style sheet	Sim. resp.	Max RT (s)	Average RT (s)
Standard	150	20.654	0.740
Standard	200	300.427	28.503
Simple	150	4.024	0.183
Simple	200	4.555	0.888

Here again we see significant better results for the Simple Style Sheet, especially when the number of respondents is 200. We can also have a look at the distribution of how fast requests are processed. The table below shows the maximum waiting times in ms for a percentage of the requests when the times are in an ascending order.

Stylesheet	Sim. Resp.	1%	5%	25%	50%	75%	95%	99%	100%
Standard	150	15	78	125	216	512	3584	8064	20654
Standard	200	1008	2288	7488	17792	35968	96896	159360	300427
Simple	150	15	62	78	110	172	608	1456	4024
Simple	200	31	63	156	624	1440	2528	3328	4555

Acceptable waiting times are around 2 seconds. For the simple style sheet these are accomplished for almost 95% of the requests when the server is tested with 200 respondents. For the standard style sheet this is only achieved for 5% of the requests.

6 Security enhancement

In our setup, respondents log in into a web portal with their own username and password. Once logged in, respondents are directed to the Blaise IS questionnaires with an integer primary key as KeyValue parameter (either GET or POST). This setup is unsafe, since the respondent can alter their own primary key value in the portal and would thus be able to fill in the questionnaire under a different identity and/or potentially see the answers of other respondents.
We came up with a simple and robust solution. In the portal, a hash value (we use the sha1 algorithm) is calculated from the concatenation of a salt string (normally bigger than 20 random characters) and the KeyValue. This hash value is submitted together with the KeyValue to a modified start page of BlaiseIS (biInterviewStarter.asp). In this start page, we added code to recalculate the hash value from the KeyValue using the same salt string and compare this value with the submitted hash. Respondents only get an active interview session if the recalculated hash matches the submitted hash.

7 Conclusion

We discovered that the XSL transformation can be a bottle neck in the performance of the system when the server load is high. In summary we can say that the simple style sheet can improve the server capacity. More respondents can be served by the same server and these respondents have shorter waiting times for a page to be generated.
The modifications to the Blaise IS system described in this paper make the system better suited for web questionnaires for large groups of respondents using limited hardware resources. Especially for large projects with relative high sample sizes, it pays off to invest in optimization of the style sheet.

8 References

[1] WCAT 6.3 – MicroSoft Corporation
http://www.iis.net/community/default.aspx?tabid=34&g=6&i=1466