

# Web application performance testing — a case study of an on-line learning application

J Shaw

---

*As the Internet continues to expand and eCommerce develops, Web applications are becoming increasingly important sources of revenue. A literature survey revealed that users are not tolerant of delays of more than 10 seconds when using applications and only 52% of Web applications meet their performance targets when launched. This paper contains a study of the performance testing of one Web application. It concludes that performance testing can predict performance problems and, if started early in the life cycle, with a close working relationship between the developers and testers, it should enable service-affecting performance problems to be avoided. Simply specifying high-performance hardware or particular software technologies did not result in good user-perceived response times in the case study.*

---

## 1. Introduction — business drivers

The Internet has evolved considerably from its origins as a resilient way of linking cold war computers. Academics used it as a means of sharing information and the invention of the World Wide Web spurred its growth into homes and businesses across the globe. Information available on the World Wide Web was originally static and unchanging and while this provided a simple mechanism for the publication of information, it was soon realised that the information would be more useful if it could be customised for the recipients. Listings gave way to search-engines, and new technology made it possible to submit forms of information that could be processed on-line and results returned — the interactive age had dawned. The forward-thinking realised that money could be made out of the World Wide Web, beginning with static advertising banners on popular sites. Advertisers then wanted to be able to target specific groups, based on their usage profile or their stated interests. Services became available too, from the purchase of books to cars, credit checking of companies, and on-line training. Commercial activity over the Internet, dubbed 'eBusiness', is predicted to grow significantly in the coming years; for example, in Europe the market for on-line training is expected to increase from \$70 million in 1998 to \$200 million in 2002 [1].

Increases in the sophistication of the services offered have been mirrored by advances in the technology delivering the services. Today a request for a page may be authenticated against a database, an advert loaded from one server, some content loaded from a local server and other content from a remote server that may be run by a third party. However, this increased sophistication can result in slow response times.

It is well documented [2] that users are sensitive to delays in response from computers and it is generally considered that delays of greater than 10 seconds are distracting and disrupt users' work patterns. On a captive system users develop strategies to cope with delay [3] — on the Internet one of those mechanisms may be to go to another site offering a similar service, which may lead to a loss of revenue [4]. Therefore to meet revenue targets it is essential that Web services both function correctly and meet users' performance expectations. Delays from applications tend to increase as the number of application users increases — adequate performance may therefore be preserved by reducing the number of simultaneous application users. This strategy may also affect the revenue potential of a service. A recent study of eBusiness application scalability in the USA revealed that 52% of applications failed to support the planned number of simultaneous users when they were first deployed [5]. The results were based on a sample group of 117 responses. As the applications were designed to support an average of 3700 simultaneous users, it is apparent that building large Web applications is not straightforward.

Analysis of the figures in the Newport study [5] reveals that 95% of the applications that were load tested early in the design-and-development cycle scaled as expected. However, for applications that were not load tested early, only 22% scaled as planned. Several organisations had used load-test tools as deployment approached but this was not effective.

'The late purchase of a tool, although beneficial for future projects, tends to only work as a reactive

‘Band-Aid’ approach to a suffering application anxious for deployment. IT groups employing this reactive approach will find themselves creating a lot of extra work for themselves as they backtrack to define, develop and execute appropriate tests for their application. Often this approach unearths unexpected application problems, causes frustration for those trying to learn a new load-testing tool under a very restricted time-frame and overall does not derive the full value from the load-testing tool. Lastly, this approach causes allotted time and budget costs to expand significantly.’ [5]

For companies like BT who are planning to deploy revenue-generating Web systems, a key question is: ‘Can performance testing help predict real problems and ensure that the systems work as planned or do they just show artefacts of the test tools?’ This paper seeks to answer this question through a case study of performance testing of the Solstra on-line training application [6] before and during its deployment within BT. The Solstra product is jointly developed, marketed and sold by BT and Futuremedia PLC.

## 2. Background

### 2.1 Performance and delay

Performance is a measure of behaviour that can be defined in a number of ways, but is often also used as a collective term encompassing several complementary definitions. For example, an operations manager may view performance in terms of an application’s availability to users. An infrastructure planning manager’s key performance indicator may be the application’s capacity, coupled with concerns of how and when to increase capacity. Another measure of performance is the amount of information lost while being communicated. In this paper, performance is defined as a measure of delay or user-perceived response time; therefore the purpose of performance testing is to discover the delay users experience in accessing a system under various load conditions.

Delay relates to other performance measures, such as capacity, as illustrated in the following example. A system with 10 simultaneous users may respond to requests in 5

seconds. With a load of 100 simultaneous users the response may be 50 seconds. The system is still responding, is still available to patient users, but the user-perceived response time (or delay introduced by the application) may be considered so slow that it has become unusable. If it is unusable it is effectively unavailable and has therefore exceeded its capacity.

### 2.2 Sources of delay for Internet services

Users experience delay from Internet services from a variety of sources (see Fig 1). The most studied source of delay is the network. This does not make performance prediction easier as the route from user to application will be dependent on the choice of Internet service provider (ISP), modem speed and line quality, as well as other factors that may be unknown to the application service provider. Some users are swift to blame slow performance on their personal computer (PC), and this can be a factor. It becomes more apparent when the content requires client processing, for example, variable width tables, frames and JavaScript code within the HTML pages. Although not so much of an issue now that the performance of PCs has increased, it can still affect users of slow machines, and is separate from the time to download the files over the modem. However, the size of the files that make up a page is a content issue; for example, studies of the Kodak homepage revealed that user complaints increased as the total page size (all frames and graphics) grew over 35 kbytes and 55 kbytes was considered unacceptably large for modem users [7]. The final category of delay comes from the application itself and may be influenced by, for example, the processing ability of the servers and the coding of the application. In this case study, the focus of attention has been these application delays.

### 2.3 Application performance test tools

To test that an application or service can support a certain load, a known load must be generated. This can be achieved manually but is expensive in both human and computer resources. Timings may also be variable and collating the results can be time consuming. A more repeatable option is the use of load generation tools. There is a choice of load-testing tools available, ranging from freeware to fully

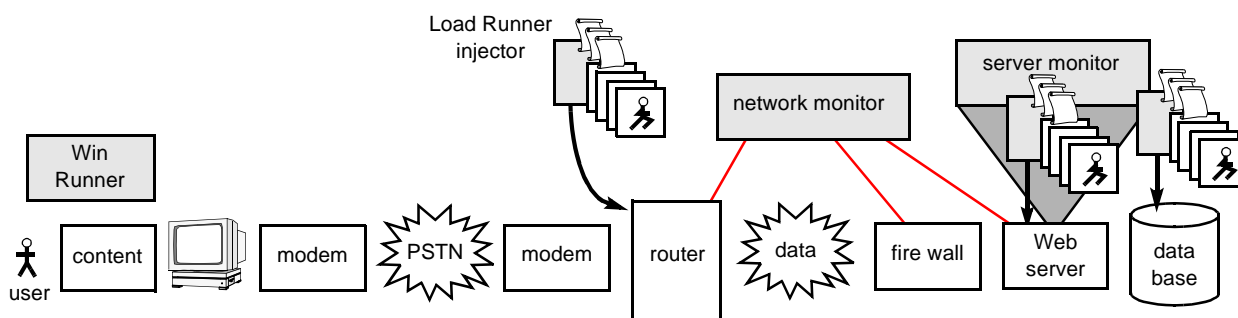


Fig 1 All the elements between the user and the application introduce delay in addition to the application response time. Load Runner can inject load and measure timings at various points in the network. The response of the servers and timings across the network can be measured using the server and network monitors. Load Runner can also be integrated with Win Runner, which then allows accurate user-perceived timings to be gathered.

supported tools with substantial purchase and maintenance charges. At one end of the market, the tools simply request pages and time how long it takes for the page request to complete. Web-Stone [8] had been used in BT, mainly to characterise Web server software on a variety of platforms, but it did not support authentication, the use of cookies<sup>1</sup> or redirection<sup>2</sup>. The more costly tools support these features and therefore allow whole business transactions to be mimicked, simulating the pauses between page requests that a human user would make. These tools (e.g. Load Runner [9] and SilkTest [10]) allow modern transaction-based Web applications to be fully tested. Load Runner was used in this case study following an evaluation which considered the functional abilities of the tools available, along with their maturity, support offered, and cost.

### 3. The process

In this case study, a Web-based on-line training application, Solstra, was studied, which is used to deliver and track the progress of training courses in BT, and is also offered commercially by BT and Futuremedia PLC. Although Solstra consists of a suite of tools and services, the main part of the system, known as the Administrator, was the only part tested using Load Runner.

<sup>1</sup> A cookie is a small file given to a client by a Web server. The client sends the cookie with each subsequent request to that server. Servers can give clients multiple cookies. Cookies can be used in a variety of ways including authenticating the user with a server, recording a user's preferences, tracking how often a user visits a server. Some test tools cannot store these for each virtual user.

<sup>2</sup> A server can respond to a page request by telling the client that the page has temporarily moved to a new address that it gives the client. The client then automatically requests the new address. The process is known as redirection and is not usually visible to the user. It can be used to ensure that a user authenticates before accessing information, so that site usage can be monitored. Some test tools treat the redirect message as an error.

The rate of change in the Web environment means that tool features can lag behind technologies that developers implement. Therefore the first stage in this project was a set of confidence tests to check that the tool, Load Runner, could be used to place a measurable load on the system. The next stage was to identify and script the key business processes of importance to the customer and Solstra users. Once the scripts had been validated, they were run individually to baseline the behaviour of the application. The final stage in the test run was to combine the scripts into realistic scenarios that mirrored the number of users performing the scripted action at a given time.

The measurement process was repeated with later versions of the software so that improvements could be quantified. In essence, exactly the same tests were used again, but some of them required reimplementing for the new environment. The results of these tests are reported in section 5.2 and are discussed in section 5.3.

The key processes/scripts for Solstra are derived from top-level menu options (see Fig 2).

The timings collected by virtual users are essentially download times for the files that are downloaded to the client. Collecting data for the same tests at different points in the network can help identify where delays are added *en route* to the user, but they are not user-perceived times. Load Runner can be integrated with its sister tool Win Runner, which actually drives the browser GUI interface and therefore allows accurate user-perceived delays to be measured, although this was not done in the case study. Knowledge of operating system performance is also important when load testing as it can help pinpoint parts of the system that are working particularly hard. In these tests

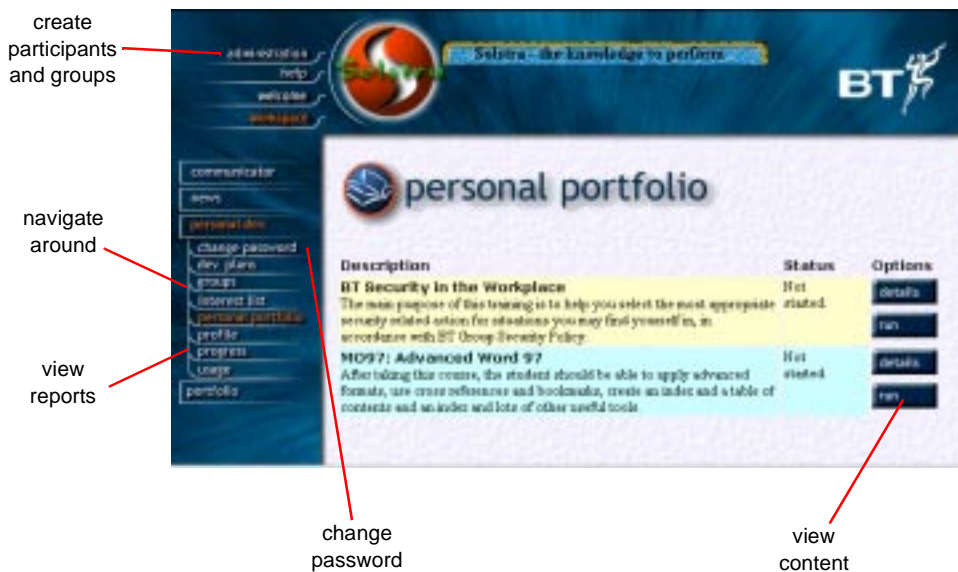


Fig 2 The Solstra administrator participant window.

system performance data was manually collected using NT Performance Monitor.

**4. Environment**

The performance tests were designed to give confidence that the initial phase of the deployment of Solstra within BT would be able to support 15 000 users. As this deployment was planned for a robust and scalable platform implemented within a managed server farm in Bletchley, the test environment for the initial tests reflected the multi-server Bletchley environment. However, an instance was also deployed in Edinburgh for a trial of Solstra and the service-surround processes; the service patches were therefore tested in a single server environment. This substantial change in the environment does make it difficult to make direct comparisons between the results, although the changes in response observed did not follow the conventional wisdom that more computing power leads to better response.

**5. Results and discussion**

This section contains a summary of the key results and a discussion that covers both the performance of the Solstra application and lessons learnt about the performance testing process.

*5.1 Key results, Solstra Version 1.4, initial release*

Viewing content

An essential function of a training system is the presentation of training material to the participants; this was assumed to be the activity that around 90% of concurrent users would be performing. In the script, each of the files is downloaded by each of the 100 virtual users, and each file is represented by a distinct mark on the graph in Fig 3. In all, this graph records the duration of 9300 downloads, the vast majority of which take less than half a second. The system log from the Web server showed very low levels of CPU usage while responding to these page requests, indicating that the Web server handles the provision of simple Web pages very efficiently.

Navigation through screens

In normal usage the content was viewed with a special viewer which prevented users from accessing the content

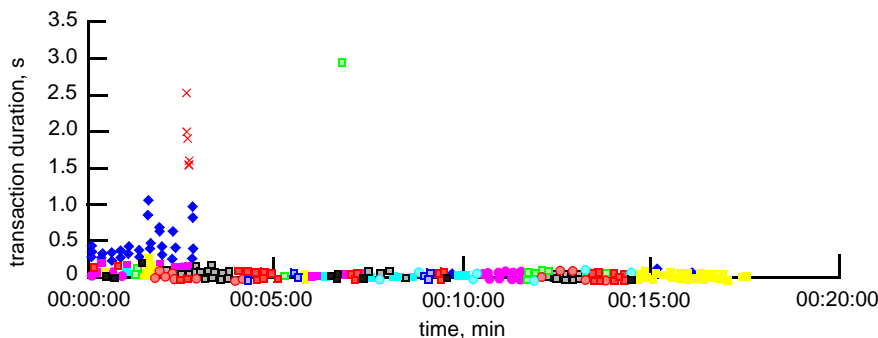


Fig 3 Times to download content files against the time in the scenario they were downloaded.

directly; participants had to navigate to content through the Solstra Administrator screens (Fig 2). The results showed that the majority of navigation pages loaded in less than 10 seconds, which is acceptable. Again the system statistics showed a light load on the server CPU during this test.

Creating participants and groups

A key administrative task was the creation of users of the system (participants and groups), so that a group of participants could be managed collectively. Figure 4 shows that, as the number of administrators increases, the total time to create a user or group (user\_create, group\_create) increases, and also that this is mirrored by an increase in the create transaction. The rise in the create-time indicates that administrators performing these tasks at the same time would experience unacceptable delays. Finally, it can be seen that the admin\_login transaction time rises rapidly at the 10-user point. While administrators would not repeatedly log on and off the system, it highlights a weakness in the application.

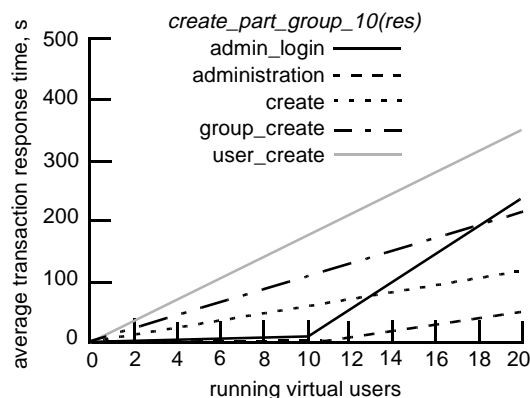


Fig 4 Time to complete the creation of participants and groups against the number of running virtual users. Note response time in 100s of seconds.

A look at the Web server load during this test reveals that, although the number of GET requests is small, the CPU load is very much higher than for serving content or navigation (Fig 5). However, there was very little activity on the database server that indicated some sort of congestion between the Solstra application and the database.

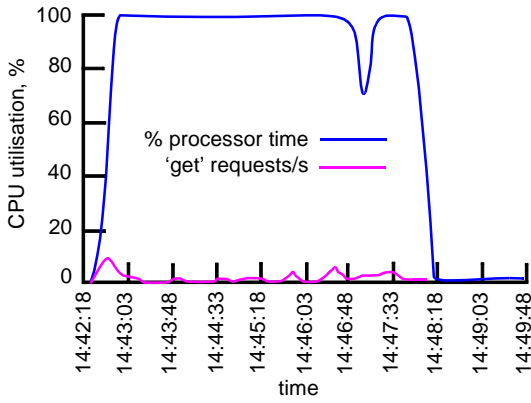


Fig 5 Web server statistics plotted against scenario time for administrators adding groups and participants.

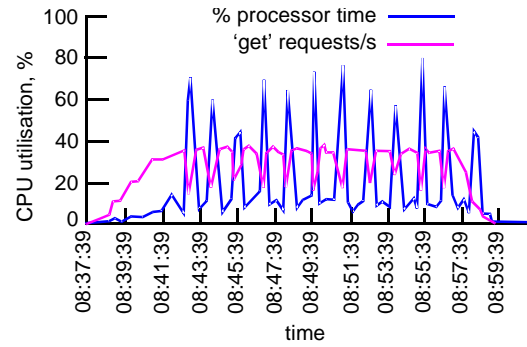


Fig 7 Web server statistics versus scenario duration for users viewing content and navigating, and administrators adding users and groups.

Combined scenario

A total of 20 administrators were simulated using the system at the same time in the previous scenario and while it highlighted a potential weakness in the application, this level of load was not realistic. The combined scenario reported in this section was representative of a real load with only two administrators using the system, one creating participants and the other groups. Bearing in mind that both these transactions have 15 seconds of think time added, and are made up of six page loads, the response of the system is on the margins of acceptability (see Fig 6).

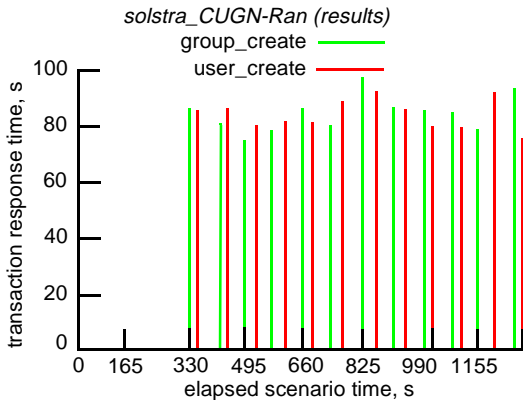


Fig 6 Transaction response time for one administrator creating participants and one creating groups while 90 participants access content and 7 participants navigate around the system.

The NT system logs for the Web and Solstra application server (Fig 7) reveal very distinct peaks and troughs in CPU, which are mirrored by the number of GET requests handled. This can be explained by considering that when the processor is busy the system becomes less able to handle GET requests. Investigation of the Load Runner transaction reports revealed that there were only two transactions that were performed at the same time as the peaks in CPU occurred, but no other times. These were group\_create and user\_create.

5.2 Key results, Solstra Version 1.4, service releases

Several service packs aimed at improving the performance and stability of Solstra v1.4 were made, the last of these (SR4) involved a rewrite of the way the application communicated with the database. Before this major change was released it was important to ascertain that it was an improvement on the service pack it replaced (SR3). As mentioned in section 4, these tests were performed on a single server model. As the combined scenario did not reveal anything that did not appear when looking at creating participants and groups, only the combined scenario is discussed here.

Creating participants and groups

It was immediately apparent that the SR3 suffered severe performance degradation as multiple users are added to the database simultaneously. From the graph of transaction time versus number of running virtual users (Fig 8) it is clear that some limit was reached at five users. The users were joining the test at the rate of 1 per min, the user\_create users running for 10 cycles and the group\_create users running indefinitely. The sharp drop in CPU after about 6 min indicated that there was a bottle-neck

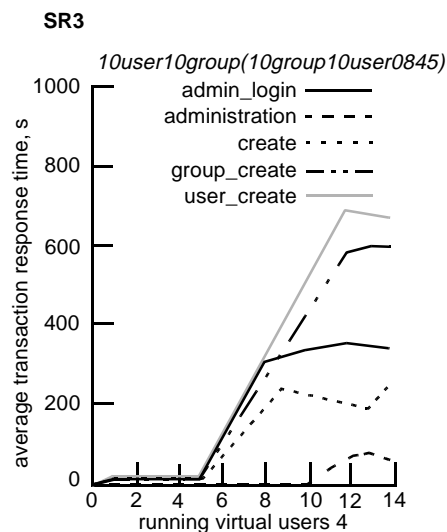


Fig 8 Transaction response time versus number of running virtual users for the SR3 release of Solstra creating participants and groups.

elsewhere in the system (Fig 9). This is an improvement on the previous version (Fig 5), where the CPU hit 100%.

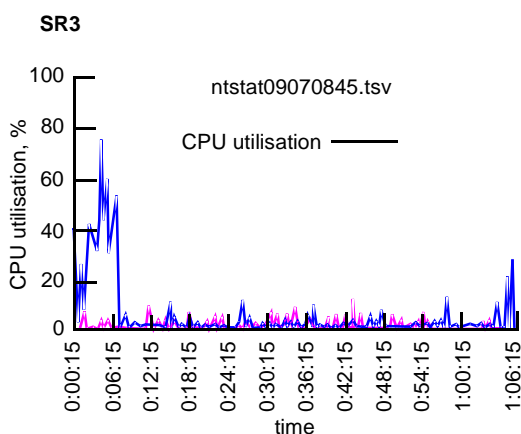


Fig 9 Application server CPU utilisation for the SR3 release of Solstra creating participants and groups.

In comparison SR4 shows no load-related drop-off within the bounds of the test (see Figs 10 and 11). Group creation took marginally longer than user creation, but, as user creation was likely to be more frequent, this was acceptable. As the transaction time more than doubles under load, users would notice an appreciable slow down, but the system does still work. The navigation through the application (administration transaction) is still very quick, the overall slow down being attributable to slower log-in and the actual creation of the account. However, these times for log-in and creation fall outside the limits of acceptable performance according to the literature [2]. As users join the test at 1-minute intervals, the CPU ramps up and stays at 100% until the scenario finishes when it returns to a nominal level — perfect behaviour. The application is now limited by the machine it is running on rather than its internal characteristics. This indicates a scalable design, in which additional demand can be accommodated by increasing the processing power.

### 5.3 Solstra Version 1.4 performance

While the behaviour of the initial version of Solstra 1.4 was interesting, the key question for the customer was: ‘Are ordinary participants affected?’ There was no specific test for this but it was possible to compare the time it took for 90% of the file downloads to complete when the system was only serving content and when it was also supporting other (administrative) user activity.

The results in Table 1 predicted that users would have been considerably affected by an increase of around 200 times for the files in the table, which were picked at random. Although the download times are within acceptable limits it was significant and undesirable that use of the application by a minority group of users can affect the response time for the majority.

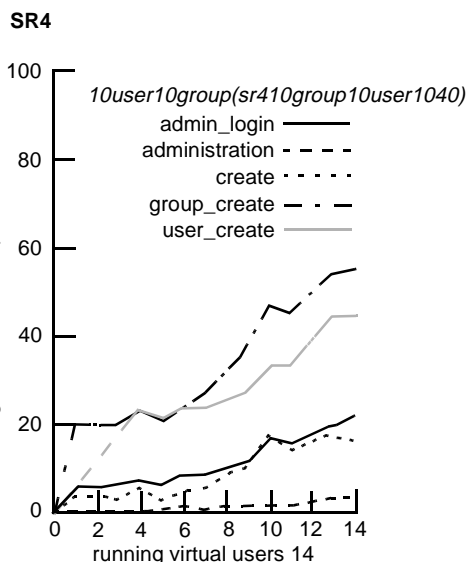


Fig 10 Transaction response time versus number of running virtual users for the SR4 release of Solstra creating participants and groups.

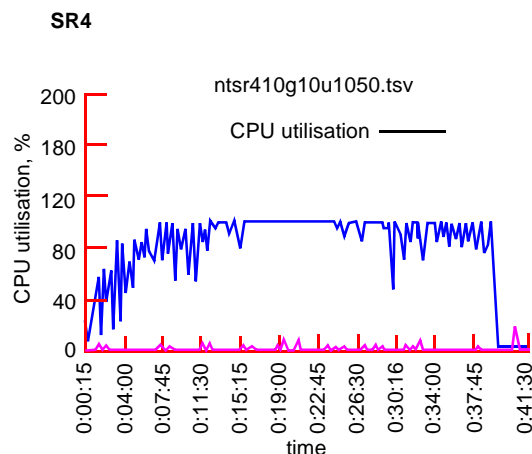


Fig 11 Application server CPU utilisation for the SR4 release of Solstra creating participants and groups.

Table 1 Comparison of content page download times when the server is only serving content and when it is also sustaining navigation through the screens and administrators creating participants and groups.

Page	90th percentile of page download times	
	Content only (s)	Combined scenario (s)
o01_220.png	0.04	1.69
o01_221a.png	0.01	1.98
o01_222a.png	0.01	2.77
o01_223a.png	0.02	2.98
o01_224b.png	0.01	4.78
o01_225a.png	0.01	5.37
o01_225b.png	0.02	4.43
o01_226a.png	0.02	4.4
o01_226b.png	0.04	5.39

Another crucial question asked by the development team was: ‘Are the tests realistic or are the results an artefact of the testing process?’ After the initial tests, Solstra

1.4 (SR3) was deployed on to a single server in Edinburgh. Participants complained of log-in delays and general sluggish response when using the service. Administrators also complained of excessive time required to create participants, change passwords and amend participant details, among other problems. In retrospect therefore the results of the performance tests had identified problem areas within the application that were visible to the users. The evidence implicated the link to the database, and the improvements under load with SR4 (changes to database communication) indicated that this was an accurate conclusion. The later results comparing the two service releases demonstrated that it was possible to identify improvements between the builds that would not have been visible to a single user.

## 6. Conclusions

### 6.1 Performance testing works, but needs planning

The case study results correlate with the Newport findings [5], that late use of performance testing does not help launch a scalable product. However, performance testing of Solstra did identify realistic performance problems, gave an accurate indication of where the main problem lay and made it possible to quantify improvements in subsequent patches (section 5.3). These events demonstrate a number of points:

- performance testing with a tool like Load Runner can generate a system activity that mimics the behaviour of real users and reveals the problems they will encounter with products before they are released into service — a testing-fix cycle before release can iron out these problems,
- it is essential to agree an approach to performance testing before the start — this must include the development and deployment as well as test teams,
- performance testing should be planned into the deployment schedule in such a way that there is sufficient time to resolve problems discovered during the performance testing.

The problems in the case studied here suggest that interfaces to databases should be given attention as soon as possible in the life cycle. As performance testing depends on a functionally reliable system, this may be difficult, but Load Runner can mimic other protocols, which may aid early life-cycle testing of critical interfaces.

### 6.2 Close relationship between development and test teams

This work demonstrated that a detailed understanding of the system was not necessary to identify areas of poor performance from a user perspective. The independent stance of a test team helps them champion the user view, but

the development team is best placed to make a detailed diagnosis of problems and provide fixes. Ensuring that a service will sustain the planned load is a joint effort between the teams and when achieved, recognition of this should be shared between the teams.

While retaining a level of independence, the test team must be seen as a key part of the project team. A football analogy can be drawn. People remember the goal scorers in a team but defenders are not well recognised. However, this does not mean that a team of well-known strikers would succeed, since defenders have different but equally essential skills. This is much the same principle as discovered by Belbin with his Apollo team studies [11], but it can be overlooked when development and test teams are selected and budgets allocated.

### 6.3 Powerful machines do not guarantee good response

The assumption that adding hardware can solve performance problems is common. Adding hardware is useful when a system has become limited by its environment. However, if a system is limited because of its own internal structure, adding hardware can be an expensive diversion. The results here demonstrate this point — the worst performance was recorded in the more powerful environment (compare Figs 4 and 8). Similarly the use of a particular software architecture, in this case Java Servlets, is no guarantee of adequate performance. Whatever the approach taken to implement a system, it must be carried through thoroughly to realise the potential benefits. Therefore, if performance problems are encountered, an understanding of what is causing the slow response should be the starting point for resolving the problem, and the development approach, architecture, usage and environment must be included in the search for a solution.

## Acknowledgements

The author gratefully acknowledges the help and support of colleagues at BT for providing the opportunity and the tools to complete the MSc on which this paper is based. Professor Jon Crowcroft, of University College London, Chris Carter and Paul Whiting, both of BT, provided much appreciated guidance during the project. Special thanks are also due to Futuremedia PLC for its collaboration with this work.

## References

- 1 Preston T (Solstra Product Manager): 'Presentation on Solstra', (1999).
- 2 Shneiderman B: 'Designing the user interface', Addison Wesley, Wokingham, especially chapter 7: 'Response Time and Display Rate', (1992).
- 3 Lansdale M W and Ormerod T C: 'Understanding interfaces', Academic Press, London (1994).

## WEB APPLICATION PERFORMANCE TESTING

- 4 Crosby A (Mercury Interactive): Interactive presentation, quoting a Forrester report saying that delays of greater than 7 seconds will lead to 20% of users going to an alternative supplier and not returning (1995).
- 5 Newport Research: 'Making E-Business Work', Newport Group, Barnstable, Massachusetts (1999).
- 6 Solstra — <http://www.solstra.com>
- 7 Neale W: 'Effective Web site design', User Interface, Boston Massachusetts (November 1997).
- 8 Web-Stone — <http://www.mindcraft.com/benchmarks/webstone/>
- 9 Load Runner — <http://www.merc-int.com/products/loadrungle.html>
- 10 SilkTest — [http://www.seguate.com/html/s\\_solutions/s\\_performer/s\\_performer.htm](http://www.seguate.com/html/s_solutions/s_performer/s_performer.htm)

- 11 Belbin R M: 'Management teams — why they succeed or fail', Butterworth Heinemann (1981).



Jonathan Shaw leads a team specialising in identifying and resolving performance problems with Internet services. He joined BT's optical devices division in 1988 following a degree in physics from Birmingham University. Subsequently he moved into the realm of computing and worked on various network and service management systems.

In recent years he has led test teams for groupware and Internet products. In 1999 he was awarded an MSc with distinction in Telecommunications from the University of London and a prize for his project work on Web application performance.