COMPARATIVE PERFORMANCE EVALUATION OF E-COMMERCE TECHNOLOGIES: A TPC-W-BASED BENCHMARKING TOOL

YUSSUF N. ABU SHAABAN* and JANE HILLSTON

ICSA, School of Informatics, University of Edinburgh.

Abstract E-commerce systems are an important new application area in which maintaining good performance under scaling workloads is crucial to business success. The TPC-W benchmark is a benchmark designed to exercise a web server and associated transaction processing system in representative e-commerce scenarios. Whilst the benchmark specifies the architecture of the system, and the form of the interactions between users, web server and underlying database, it does not stipulate the supporting technology used to communicate between the web server and the database. In this paper we describe a tool which has been developed to allow comparative benchmarking of different technologies with minimal re-implementation. The tool implements the TPC-W specification in a modular fashion which allows different application servers to be inserted and experimented upon. The use of the tool in the comparative benchmarking of Java Servlets and PHP is demonstrated

Keywords Software Tools, Software Performance, Performance Modelling, Measurements Techniques, Workload Modelling and Characterization.

1 INTRODUCTION

In recent years e-commerce systems have become one of the most noticeable manifestations of the Internet phenomenon. Although there are no widely accepted definitions, such systems are generally deemed to provide facilities for commercial transactions to take place between remote participants. In particular, three categories of e-commerce systems are identified [3]: *Electronic Markets*, *Electronic Data Interchange (EDI)* and *Internet Commerce*. For the remainder of this paper we will treat "*e-commerce*" as synonymous with *Internet commerce*.

In many e-commerce systems, timely behaviour is crucial in order to maintain the site owner's competitive edge: poor performance can quite literally translate into lost revenue [1]. However, little systematic work has been done on analysing the performance of such systems and their supporting technologies. In this paper we describe a tool which aims to provide a framework in which different e-commerce programming technologies can be easily benchmarked against the standard TPC-W benchmark [11].

The TPC-W benchmark has been developed by the Transaction Processing Performance Council (TPC) as a response to the rise of e-commerce systems. It specifies the behaviour of an on-line bookstore, including many of the elements commonly found in e-commerce applications: a web-site supported by a web serving component which can present both static and dynamic web pages; a relational database which is accessed from the web server to provide transaction processing and decision support. Moreover the benchmark also specifies emulated remote browsers for different classes of customers, providing the workload on the system.

Our tool implements the TPC-W benchmark, in Java, as a framework for the investigation of application server technologies. The application server sits between the web server and the database and provides the business logic as well as the interface between the manipulation and the presentation of the data. Whilst the benchmark places constraints on the application server it does not stipulate how it should be implemented. Available technologies for this module of the system include CGI, ASP, Java Servlets, JSP and PHP. Furthermore, the tool incorporates support for experiment design, allowing the user to replicate runs of the benchmark and compare the results obtained with different workload mixes. A detailed description of the tool design can be found in [8].

The remainder of the paper is structured as follows. In Section 2 we review the role of the application server within an e-commerce system and the different technologies currently available. In Section 3 we describe the TPC-W benchmark and in the following section we show how this is captured within the design of our tool. Section 5 describes the implementation of the tool. Results of using the tool in evaluating the scalability of Java Servlets and PHP are given in Section 6. Finally in Section 7 we give some conclusions.

2 APPLICATION SERVERS

The application server is a vital component of an e-commerce system: it is the software component that

*Corresponding author, DIRC, supported by EPSRC studentship 00317428 contains the business logic of the system. It is responsible for receiving HTTP requests via the web server from clients and executing the business functions associated with each of them. This can involve interacting with database servers and/or transaction servers. The application server also has the responsibility of dynamically building web pages, formatting database query results in HTML, to be sent back to clients.

Today, many programming environments exist for implementing application servers, fulfilling the need for dynamic web-site generation and allowing the connection between web front-ends and databases. These include the Common Gateway Interface (CGI) [4], Microsoft's Active Server Pages (ASP) [5], Personal Home Pages (PHP) [10], Java Servlets [7] and Java Server Pages (JSP) [6]. Currently in the tool we offer implementations of Java Servlets and PHP, and we give a brief description of these below:

Java Servlets Java Servlets [7] are server side Java code that runs on a server to answer client HTTP requests. Servlets make use of the Java standard extension classes in the packages `java.servlet` and `javax.servlet.http`. Since servlets are written in the highly portable Java language and follow a standard framework, they provide a means to create sophisticated server extensions in a server and OS independent way. When a servlet program is called for the first time, it is loaded into memory. After the request is processed, the servlet remains in memory and will not be unloaded from memory until the server is shutted down. Servlets offer excellent connectivity with many databases through the use of Java's JDBC package. The output of HTML in a servlet environment is an issue, as the servlet is required to output all HTML internally. This requires complicated output statements to handle the output of the entire HTML content, as well as the code for the rest of the application.

Personal Home Pages Personal Home Pages (PHP) [10] is a server-side, cross-platform, HTML embedded scripting language. It was developed late in 1994 by Rasmus Lerdorf to keep track of visitors to his on-line resume. Since then, it has undergone several changes with two versions released, PHP3 and PHP4. Currently, PHP is shipped with a number of commercial products such as Stronghold web server and RedHat Linux. PHP provides a programming approach similar to VBScript of ASP [5], but its broad support for databases gives it an edge over VBScript. Its code can be embedded directly into a HTML page and executes on the server. PHP modules are lightweight and speedy and have no process creation overhead.

3 TPC-W BENCHMARK

The TPC-W benchmark [11] has been developed by the Transaction Processing Performance Council (TPC), a consortium of system and database vendors. Historically, TPC has specified standard benchmarks for evaluating the performance of both transaction processing and decision support database systems. One of its latest benchmarks is TPC-W, an e-commerce-specific benchmark. It specifies the behaviour of an on-line bookstore, including the three main components of an e-commerce application: remote browsing, web server and database server. TPC-W's remote browsing specifications are described next. This is followed by an overview of the web and database server components. Finally, TPC-W's performance metrics and measurement intervals specifications are discussed.

Remote Browser Emulator A main component of the TPC-W benchmark is the Remote Browser Emulator (RBE), which is a specification for a set of Emulated Browsers (EBs). EBs simulate the activities of concurrent web browsing e-commerce users, each autonomously traversing the bookstore web pages by making requests to a web server. Each EB can represent one of three classes of users: a customer, a new user or a site administrator. As described in the next paragraph, TPC-W defines 14 web interactions which can be requested by an EB. During its lifetime, an EB requests a sequence of these web interactions moving from one interaction to the next, in the same way that a web browsing user navigates a site clicking one hypertext link after another. TPC-W specifies the next possible navigation options that can be requested by an EB on completion of each of the web interactions defined in the benchmark. Threshold integer values between 1 and 9999 are specified for each navigation option. To select its next request, the EB generates a random number, from a uniform distribution between 1 and 9999. It then selects the navigation option for which the threshold is equal to, or most immediately greater than the random number. The EB spends a random period of time (*Think Time*) sleeping between subsequent web interactions. This emulates the user's think time and is generated from an exponential distribution specified in TPC-W. User-specific information must be maintained in an EB, possibly including session tracking details and customer identification.

Web and Database Servers The TPC-W benchmark defines 14 web interactions to be supported by a web server component which are: Home, Shopping Cart, Customer Registration, Buy Request, Buy Confirm, Order Inquiry, Order Display, Search Request, Search Result, New Product, Best Seller, Product Detail, Admin Request and Admin Confirm. These interactions vary in the amount of server-side processing they need. Some require dynamically generated HTML pages and one or more connections to a database. Others are lightweight, requiring only web serving of static HTML pages and images. For each web interaction, TPC-W specifies its input requirements, processing definition, response page definition and EB navigation options which are the set of web interactions that can be selected by the EB on completion of the interaction.

Session tracking is vital to any e-commerce application. Some method is needed to retain information such as shopping carts from one HTTP request to another. TPC-W suggests two techniques for session tracking which are URL-rewriting and cookies [2].

The TPC-W benchmark defines the exact schema used for an online bookstore database consisting of eight tables: *item*, *customer*, *address*, *order*, *order line*, *credit card transaction*, *author* and *country*. A *scale factor* is also defined, that is the size of the *item* table. The size of the database depends on the number of EBs that will be used as a workload and the *scale factor*. TPC-W specifies database table sizes as follows:

- Item: Scale Factor.
- Customer: 2880 * Number of EBs rows.
- Country: 92 rows.
- Address: 2 * Number of customers rows.
- Orders: 0.9 * Number of customers rows.
- Order_line: 3 * Number of orders rows.
- Author: 0.25 * Number of items rows.
- cc_xacts: 1 * Number of orders rows.

Performance Metrics and Measurement Intervals TPC-W defines one primary performance metric which is throughput, measured as the number of completed web interactions per second (WIPS). Three distinct measurement intervals are specified by TPC-W: shopping, browsing and ordering. They are distinguished by the ratio of browsing-related web pages visited to ordering-related web pages visited during the measurement interval. The shopping interval is intended to reflect a shopping scenario, in which 80% of the pages visited are related to browsing and 20% are related to ordering. In a browsing interval, ordering pages visited go down to 5% whereas in an ordering interval the ratio of browsing and ordering is even.

4 DESIGN OF THE TOOL

Our tool allows comparative benchmarking of e-commerce programming technologies. In designing the tool, a number of high-level design objectives were emphasised. These are discussed below. In the following we present an overview of the design. A detailed design description can be found in [8]

- It was felt important that the tool should provide experiment design features allowing the user to specify a number of experiments. For each experiment, factor level combinations such as size of workload, size of store and measurement interval type can be specified. The number of replications for each experiment can also be specified.
- Another key objective was to provide performance metrics to assist e-commerce technology performance evaluators. Metrics such as response time frequency distributions of web interactions, overall response time and overall throughput are analysed and presented graphically.
- Ensuring that the analysis of experimental results have minimal overhead was an objective from early stages in the design, thus preserving the realistic nature of the simulation. This resulted in a local data collection strategy in the design. Experimental data results are maintained locally by different parts of the system. Results are gathered at the end of each experiment run from different parts of the system for analysis.
- Realistic e-commerce modelling was an important design criterion. This was ensured by adopting TPC-W as an e-commerce model. Not only different e-commerce site components are represented faithfully; the design also includes realistic workload generation capabilities based on TPC-W's e-commerce access patterns.
- Developing a flexible, extendable tool was a major concern. This led to a modular design where incorporating a new programming technology for benchmarking just involves adding a new module to the tool. In addition, modular design allows for the easy extension of the tool to benchmark other e-commerce components such as security protocols, session tracking techniques, etc.

Figure 1 illustrates the main modules of the tool. The RBE module is responsible for generating and maintaining the workload on the Web and Database Servers. Its design is based on the TPC-W benchmark as described in Section 3. The Web Server contains the application server which represents the application code for implementing the 14 web interactions specified in the TPC-W benchmark (see Section 3). The Database Server represents the persistent storage component of the TPC-W e-commerce model. The final main module in the design is the Control Unit which provides experimental design features, data gathering and analysis. It is also responsible for controlling the setup and maintenance of experiments on various parts of the tool. The Control Unit module is described next, this is followed by a description of RBE and finally the Web and Database Servers are described.

Control Unit The Control Unit is the central component of the benchmark. In addition to providing a GUI to the user, it is responsible for setting up and maintaining the running of experiments. It is also responsible for gathering experiment data results from different parts of the system for analysis and presentation. Figure 2 illustrates the main sub-modules of the Control Unit. The ControllerGUI provides a GUI allowing the user to input experimental design details. The Experimental Design component is responsible for holding the experimental plan of the user. It informs the Control Unit of the experiment design details needed at each stage of the simulation. As the user is allowed to specify a number of experiments which are run sequentially, the Experimental Design component consists of a set of Experiment objects, each holding the design details for one experiment including: Number of replications, workload size (number of EBs to instantiated), measurement interval type and a scale factor contributing to the determination of the bookstore database size (see Section 3). Experimental data gathering and analysis is the responsibility of the Result Analysis Unit. It consists of a set of Experiment Results objects holding the results for each experiment executed in the simulation. Each Experiment Results object is linked with an Experiment object and contains a set of Run objects holding the results of each experiment run. During an experiment run, experimental data is recorded locally by different parts of the

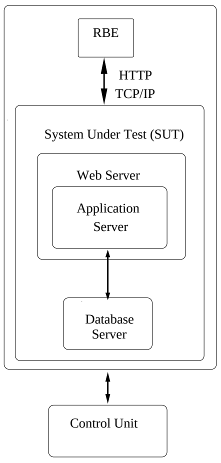

Figure 1: Tool Overall Design

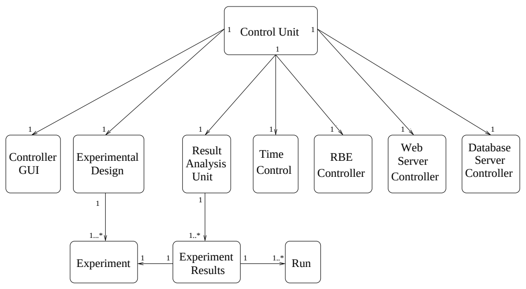

Figure 2: Control Unit

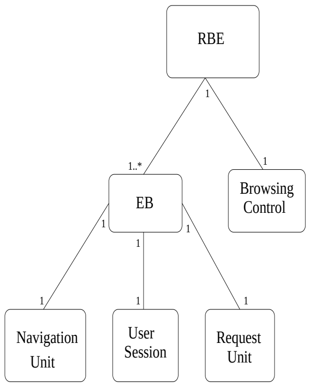

Figure 3: RBE

system minimising data recording overhead. On completion of the run, the Result Analysis Unit receives a request from the Control Unit to gather the experimental data which are then stored in a Run object. Data analysis is done after the completion of each experiment. Performance metrics provided include overall average throughput, overall average response time and individual interactions average response time (see Section 6). Time Control is the central timing component of the tool. It distributes timing information to all parts of the system. The Control Unit also includes three controller components, RBE Controller, Web Server Controller and Database Server Controller for controlling the RBE, Web Server and Database Server respectively. Instructed by the Control Unit, these components configure and control the running of experiments on the parts of the tool they are responsible for. Controlling an extension to the tool requires only adding a new controller to the Control Unit.

RBE RBE is the component responsible for driving the tool workload. As shown in Figure 3, it includes a set of EBs which emulate web browsing e-commerce users requesting web interactions from a web server as specified in the TPC-W benchmark (see Section 3). Its Navigation Unit emulates the user's navigational behaviour. The decision on which navigation option to select next is based on the current web interaction just completed and its EB navigation option thresholds, specified by TPC-W (see Section 3). The Navigation Unit generates a random Think Time period which is spent sleeping between web interaction requests. The EB User Session sub-module is designed to emulate the maintenance of e-commerce user-specific information in a web browser including session tracking details and customer identification. Cookies was the method adopted for session tracking. The sending and receiving of HTML content via HTTP and TCP/IP is emulated by the Request Unit sub-module. It is responsible for forming HTTP requests for the different web interactions specified in TPC-W. It is also responsible for maintaining HTTP/TCP/IP connections with the web server and associating these connections with User Sessions. In addition, the Request Unit log data statistics about web interactions requested and completed successfully including the starting and finishing time of a web interaction, its type and how many times it has been requested in an experiment run.

EBs are created and maintained by the Browsing Control component. It receives instructions from the Control Unit (via its RBE Controller) on the measurement interval type and the number of EBs required for each experiment. When creating an EB, Browsing Control provides it with the type of user it presents (customer, new user, or site administrator) which is selected randomly. The measurement interval type is also provided to the EB. Browsing Control also informs EBs of the start/end of a measurement interval and/or an experiment run (as ordered by the Control Unit). Data recorded by the Request Unit is collected by Browsing Control to be passed to the Control Unit.

Web and Database Servers The Web and Database Servers represent the system tested by the workload. The Web Server is responsible for serving HTTP requests for the 14 dynamic and static web interactions specified in TPC-W (see Section 3). It contains the application code for the dynamic web interactions, implemented by the programming technologies currently benchmarked by the tool. The HTML code for the static web interaction and images for different interactions also reside in the web server.

The Database Server contains the *bookstore* database with a schema following exactly the one specified in TPC-W (see Section 3). The Database Server Controller of the Control Unit populates/de-populates the database according to TPC-W specifications.

5 IMPLEMENTATION

In this section we discuss the tool implementation issues. First, the decision to implement the tool in the Java language is discussed. The use of Apache as the web server of the tool is then described and the choice of MySQL to manage the bookstore database is discussed. This is followed by a description of the support provided for e-commerce programming technologies.

Java (SDK 1.4.1) is used to implement the tool. Being a pure object-oriented programming language, Java ensured the realisation of the tool's modular design. The implementation in a portable programming language as Java resulted in a platform-independent tool which can be deployed on machines of different architectures across a network. Java's `java.net` package, with its URL and socket classes, provided a neat implementation of the interactions between the EBs and the Web Server components (see Section 4). The reliable `System.currentTimeMillis()` method, which is part of Java's `java.lang` package is used to get timestamps from various parts of the system needed to produce performance metrics. `System.currentTimeMillis()` is quick with almost no overhead, thus enforcing the realistic nature of the simulation.

A web server is needed to serve HTTP requests and host the application server (see Section 4). The Apache web server (Version 1.3) is used. It is a freeware web server and is the choice of the majority of active site developers. A survey by Netcraft [9] on October 2001 concluded that 61% of the active sites they monitor use Apache. The way Apache is designed was another reason for choosing it. It is built around an API which allows third-party programmers to add new server functionality. Everything in Apache is implemented as one or several modules, using the same extension API available to third parties. Thus, extending the tool to provide support for a new e-commerce programming technology requires implementing a new Apache module and incorporating it using Apache's API.

A Database Management System (DBMS) is needed to host the bookstore database (see Section 4). It was decided to use MySQL (Version 3.23). MySQL is a popular DBMS and works well with Apache for different e-commerce programming technologies.

Currently, the tool provides application server implementations in two e-commerce programming technologies; Java Servlets [7] (using `mod_jserv` apache module) and PHP [10] (using `PHP` apache module). Providing benchmarking support for another e-commerce programming technology requires only finding/implementing a supporting Apache module and implementing the application code for that technology.

6 RESULTS

To demonstrate the capabilities of the tool, experiments were designed and implemented to compare the scalability of the Java Servlets and PHP application server implementations. The effect of varying the size of the workload on overall average throughput, overall average response time and individual web interaction average response time was examined.

Experiment Design The technologies involved in the experiments performed were Java Servlets and PHP. The size of the workload was varied by doubling the number of EBs at each stage with a minimum value of 1 EB and a maximum value of 128 EBs. The scale of the database used was 1000. The total time interval of each experiment run was 400 sec. The tool allows for a *Rump up* period of 1/4 * total time interval (100 sec in this case) for EBs initialisation. The measurement interval (during which measurement is recorded) takes the remaining 3/4 of the total time period (300 sec in this case). The measurement interval type used was Shopping reflecting a shopping scenario (see Section 3). Finally, each run was replicated 3 times. In all experiment runs, the workload generator RBE, the web server, the application server and the database server were running on a GenuineIntel Pentium III, 1 GHz, 265MB Linux Dell Machine.

Workload Effect on Throughput Figure 4 summarise the results obtained from examining the average throughput when the number of EBs is varied between 1 and 128. The average throughput is considered as the average number of web interactions completed successfully per second during the measurement interval. Figure 4 illustrates two points. Firstly, it can been seen that both Java Servlets and PHP scale well up to a workload size of 64 EBs before which the average throughput starts to degrade. Secondly, one can argue that Java Servlets scales slightly better as it degrades at a slower rate than PHP.

Workload Effect on Response Time Results obtained from varying the size of the workload and recording average response time are shown in Figure 5. The response time of a web interaction is considered as the time elapsed from the last byte received by the EB to compute a web interaction until the first byte sent by the EB to request the next interaction. Average response time is the total response time of all web interactions completed successfully divided by the number of these interactions. It can be seen from Figure 5 that the average response time starts to increase almost exponentially when the size of the workload increases above 64 EBs. Again, Java Servlets scale better with a slower rate of average response time increase. From Figure 4 and 5, one could argue that optimal performance is achieved at the 64 EBs workload, with maximum throughput and good response time

Workload Effect on Individual Web Interactions Response Time Figure 6 shows the average response time for the Shopping Cart interaction as an example of a dynamic web interaction. The average response time of the static interaction Search Request is illustrated in Figure 7. The average response time of an individual interaction is computed by totaling the response time of all occurrences of that interaction and dividing that by the number of times the interaction was requested and completed successfully. Again, it can be seen that the performance degrade point is when the workload size exceeds 64 EBs with Java Servlets degrading at a slower rate in both the static and dynamic interaction examples. It can also be noticed that the Shopping Cart Interaction (a dynamically built HTML page) average response time degrade faster than the Search Request Interaction (a static HTML page).

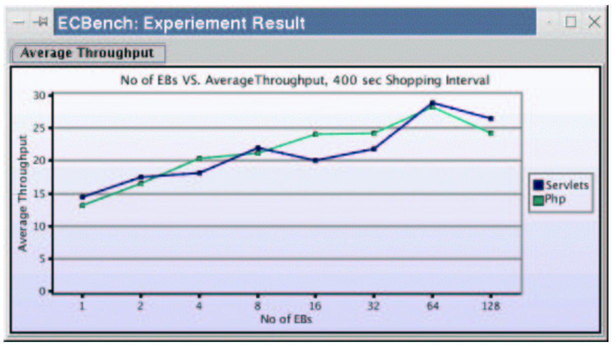

Figure 4: Overall Average Throughput

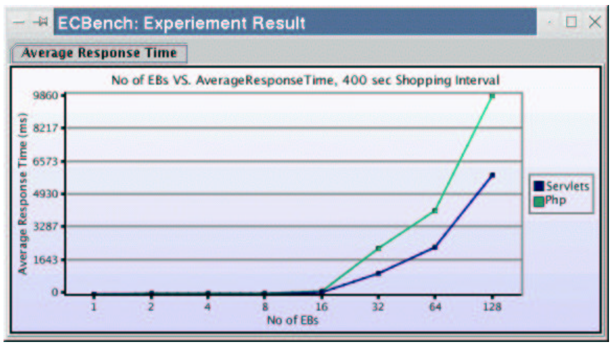

Figure 5: Overall Average Response Time

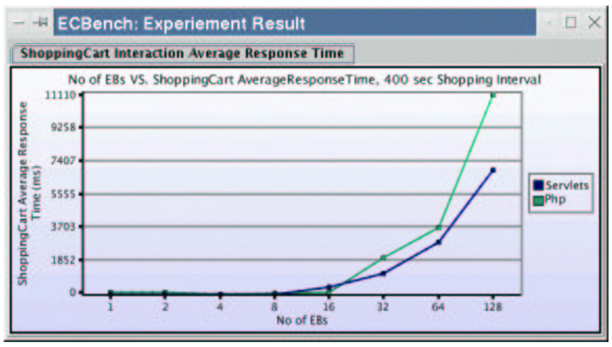

Figure 6: Shopping Cart Avg. Response Time

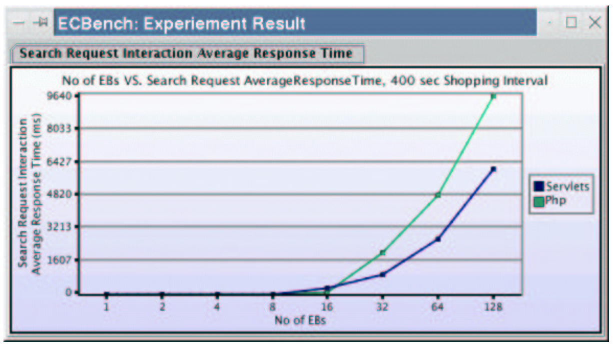

Figure 7: Search Request Avg. Response Time

7 CONCLUSIONS

A tool to allow the comparative benchmarking of e-commerce programming technologies has been developed. It is based on the TPC-W benchmark, ensuring the realistic modelling of e-commerce components. It provides a set of experiment support features allowing the user to design a set of experiments to be run sequentially by the tool. For each experiment, a number of replications can also be specified. A set of e-commerce relevant performance metrics are analysed and presented with minimal overhead on the simulation. This was achieved by adopting a local experimental data recording strategy during experiment runs. Data is gathered and analysed on completion.

The modular design of the tool widens the scope of further work. New modules can be incorporated to benchmark other programming languages/technologies. The RBE module can be enhanced to provide robots workload [12] in addition to remote browsers. The tool can also be extended with small modifications to allow for the benchmarking of other e-commerce components such as session tracking mechanisms, security and payment protocols.

References

[1] Menasce D. A. and Almeida V. A. *Scaling for E-Business: Technologies, Models, Performance, and Capacity Planning.* Prentice Hall PTR, 2000.

[2] Ince D. *Developing Distributed and E-commerce Applications.* Addison Wesley, 2002.

[3] Whiteley D. *e-Commerce: Strategy, Technologies and Applications.* McGraw Hill, 2000.

[4] Kruse M. `http://mkruse.netexpress.net/info/cgi.` Technical report.

[5] Microsoft. `http://msdn.microsoft.com/library/psdk/iisref/aspguide.htm.` Technical report, Microsoft.

[6] Sun Microsystems. `http://java.sun.com/products/jsp/whitepaper.html.` Technical report, Sun Microsystems.

[7] Sun Microsystems. `http://java.sun.com/products/servlet/whitepaper.html.` Technical report, Sun Microsystems.

[8] Abu Shaaban Y. N. and Hillston J. A TPC-W-based tool for benchmarking e-commerce programming technologies. In *Proc. 18th UK Performance Engineering Workshop, Glasgow, July 10-11, 2002.*

[9] Netcraft. http://www.netcraft.com/survey. Technical report, Netcraft.

[10] Bakken S. S., Aulbach A., Schmid E., Winstead J., Wilson L. T., Lerdorf R., Zmievski A., and Ahto J. http://www.php.net/manual.en. Technical report, The PHP Group.

[11] TPC. http://www.tpc.org/tpcw. Technical report, Transaction Processing Performance Council.

[12] Almeida V., Menasce D., Riedi R., Peligrinelli, Fonseca R., and Meira W. Analyzing robot behavior in e-business sites. In *Proc. 2001 ACM Sigmetrics Conference, June 16-20, 2001.*

Yussuf N. Abu Shaaban is a Ph.D. student in ICSA, School of Informatics, University of Edinburgh. He is part of the EPSRC-funded *Dependability IRC*. His research interest is performance evaluation of e-commerce systems.