Association for Information Systems
AIS Electronic Library (AISeL)

PACIS 2001 Proceedings

Pacific Asia Conference on Information Systems (PACIS)

December 2001

E-commerce Architecture Evaluation Through Stress Test

Young-Eun Lee
Hankuk University of Foreign Studies

Jong-Soon Park
Seo Il College

E-commerce Architecture Evaluation Through Stress Test

Young Hwan Lee
Hankuk University of Foreign Studies, Seoul, Korea

Jong Soon Park
Seo Il College, Seoul, Korea.

Abstract

Of critical importance to the success of any e-commerce site are the two factors: rapid application development and quick response time. A three-tier architecture composed of presentation layer, business layer, and data access layer emerges to allow rapid changes in user interface, business logic, and database structures. Too often, such a logical three-tier architecture is considered as requiring a three-tier physical architecture: Web server, application server, and database server running on separate computers. Contrary to the common belief, a Web stress test reveals that the three-tier logical architecture implemented on a two-tier physical platform guarantees a quicker response time due to the reduction in cross-machine communications. The stress test was conducted in a laboratory environment: the e-commerce traffic was artificially generated from a number of client computers, using scripts. It remains to be seen if the same conclusion could hold in a real environment. If so, the stress test would emerge as a tool to verify the effectiveness of e-commerce architecture, the validity of which is critical to the success of any e-commerce Web site.

Keywords: e-commerce architecture, Web stress test, capacity planning, n-tier architecture

1. Introduction

Web sites need a system architecture that would allow quick adjustments to user interface and functionalities, as well as a quick response time. As business environment changes, Web pages, business logic, and database structures need to be changed accordingly. Web masters are expensive to hire, and in case they quit, their replacements need to pick up the maintenance tasks quickly. Unless there is a well-structured software architecture in place, it is difficult to modify or enhance the system as business or personnel changes.

Customers on the Web are increasingly demanding a faster response time and a higher availability from e-business sites (Lee 2000, Reichheld and Schefter 2000) than before. Failure to provide high-quality services results in lost “eyeballs”(Menasce and Almeida 2000), hence lost revenue and a drop in the company’s stock price. There exists even an

"eight-second rule": an unsubstantiated but widely held belief that after eight seconds of waiting for a Web page to open, a customer becomes impatient and will abandon the site.

The quality of service at Web sites depends on many interrelated factors, such as hardware architecture, network capacity and software structure. Oftentimes the focus of development activity is centered on the software architecture and features. However, equally important in a Web application is the network architecture. Many well-designed applications can fail miserably on the Internet if they are not deployed and operated correctly.

Hardware architecture, software architecture, and network architecture were considered as belonging to the domain of computer scientists. However, as e-business moves to the core of a company's success, CIOs need to evaluate alternative architectures and select the best one. Failure to do so would invite expensive trial and errors. Time has come for the CIOs and MIS professionals to embrace architecture evaluation wholeheartedly: simply adding more servers can not become a solution since bottom-line impacts are critical to many Web sites. The primary purpose of this paper is to make the following arguments: (1) Web site architecture is important; (2) Stress test can be an effective tool to evaluate alternative architectures; (3) Web masters and MIS professionals should know how to use the tool to evaluate alternative architectures.

As its name implies, Web stress test artificially generates a heavy traffic to a Web site and shows if the site can handle the traffic with a satisfactory level of response time. Or given a response time constraint, the stress test can be used to calculate the maximum number of simultaneous hits at a site. Web applications are developed based on a software architecture, servers are deployed under a hardware architecture, and connections are made to Internet under a network architecture. Though a capacity planning precedes the creation of the hardware and the network architectures, a stress test is needed to verify if indeed the various components fit together. Or the stress test can be used to identify the areas of improvement, since the HTML page requests are directed to specific ASP pages on the various servers.

Architecture in this paper implies a conceptual design using graphic symbols. As the saying goes, a picture is worth one thousand words. While there is no standard, universally-accepted definition of the term architecture, there is no shortage of them either. Software architecture forms the backbone for building successful software-intensive systems. It represents earliest design decisions that are subject to change as the project proceeds. It works as a communication vehicle among the stakeholders of a system, and addresses four quality attributes: performance, reliability, modifiability and security. A system's quality attributes are largely permitted or precluded by its architecture (Kazman et al. 2000).

Another useful definition of software architecture is that it captures what a software system is

designed to do and how that system's components are meant to interact with each other. The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them (Bass et. al. 1997). Still another definition: software architecture is the set of significant decisions about the organization of a software system, the selection of the structural elements and their interfaces by which the system is composed (Booch et. al. 1999).

Why should an organization analyze its software or system architecture? Because it is a cost-effective way to mitigate the substantial risks associated with implementing a system. Complex software systems need to be modifiable, and at the same time, need to guarantee a good performance. In addition, they may need to be secure, scalable, and reliable. For any particular system, how to determine these quality attributes - modifiability, security, performance, and reliability? Can a system be analyzed to determine these desired qualities? How soon can such an analysis occur? How do you know if the software architecture for a system is suitable without having to build the system first?

Experience has shown that the quality attributes of large software systems are primarily determined by the system's software architecture. In such a system, the achievement of quality attributes depends more on the overall software architecture than on code-level practices such as language choice, detailed design, algorithms, data structures, testing, and so forth (Kazman et. al. 2000). It is therefore a critical risk mitigation measure to try to determine, before a system is built, whether it will satisfy its desired qualities. Problem is that architecture evaluation has to be heuristic, that there is no scientific way to prove the validity of a software architecture before implementing the architecture. Once the software components are developed and deployed according to a particular architecture, it is possible to conduct the stress test and change the configuration, hence change the architecture itself. In this sense, we can say that a Web stress test is an effective way to evaluate various architectures.

2. Three-tier architecture

Consider a typical B2C e-commerce site like Amazon.com where the following sequence of events occur to consummate a transaction:

1. User logs in.
2. User selects a product category.
3. User selects a product, puts it into a shopping cart, and repeats the process until satisfied.
4. User selects a delivery method

5. User provides a credit card number.
6. The site fulfils the order: deliver the products ordered to the customer.

A three-tier architecture as in Figure 1 emerges to construct a Web site that guarantees a maximum modifiability and a minimum response time (Silva and Edwards, 2000)

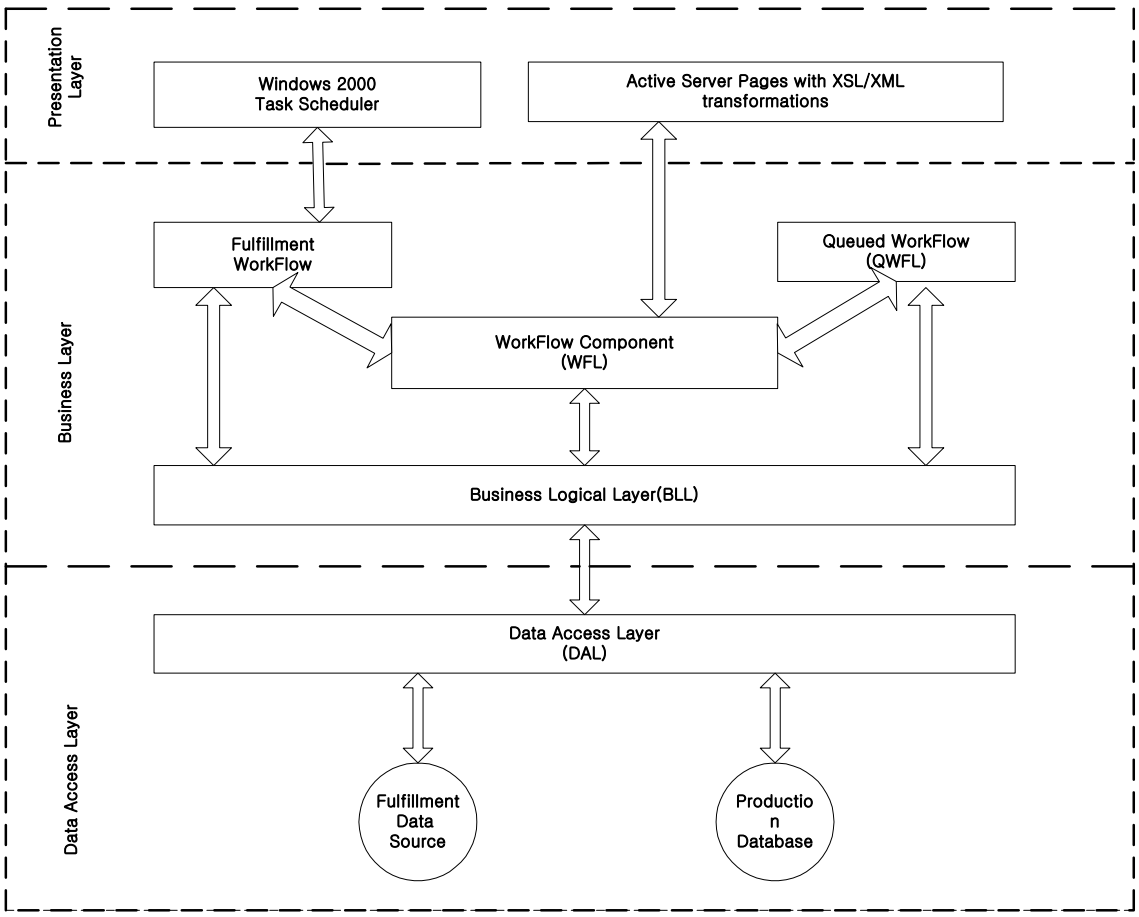

Figure 1. Logical three-tier architecture

The advantage of factoring an application into the three logical layers is to write modular, reusable, and maintainable code more easily than to write a monolithic Web application. For a Web site handling millions of transactions per day, with usage peaks of thousands per second, delaying costly operations can vastly improve response time. Queued operations free up the threads in Web server, so they can respond to more requests instead of waiting for costly synchronous operations to complete (Silva and Edwards, 2000). However, it is not an easy task to determine which parts do not require an immediate user feedback and can be queued instead. Database operations are somewhat expensive, but credit card payment authorization is very expensive. As we know from using our credit cards at restaurants, it can take a few seconds to get the approval. When all of a Web server's worker threads[1] are busy

[1] Microsoft's IIS(Internet Information Server), the software that runs a Windows 2000 or NT Web server, uses a pool of 25 worker threads.

servicing payment authorization requests, the site's response time would degrade.

In the sample Web application Duwamish Online[2] developed by Microsoft engineers, Queued Component functionality is extensively used for the order pipeline. When a customer clicks Buy, the presentation layer passes the XML-encoded order information to a local workflow component. The local component invokes a remote queued workflow component and executes a ProcessOrder method. All order processing is performed by the remotely hosted queued workflow component and is entirely out-of-band with Web server. Order processing includes inserting the sale and payment information into the database, authorizing the credit card purchase, preparing order data for fulfillment, and e-mail notification.

The fulfillment subsystem is responsible for keeping the inventory on hand in the warehouse, boxing orders, and shipping them to the customers. The fulfillment system runs as several scheduled operations, which make calls into the fulfillment workflow component to send the purchase order to the fulfillment subsystem responsible for updating order status and inventory. The fulfillment workflow is integrated with other components and uses the business logic and data access layers to perform database operations in the order tables, as well as in fulfillment database.

Business logic layer is responsible for implementing the company policies on credit limit, discounts, one-to-one marketing based on data-mining of POS data, etc. Interaction with ERP(Enterprise Resource Planning) subsystem can be handled in the context of business layer. For this reason, business layer should be treated as a separate entity to allow an easy modification of business policies.
There are two ways to implement the logical three-tier architecture. The first one is a physical two-tier architecture as in Figure 2.
Here, the business logic layer resides in Web server physically. In practice, multiple Web servers are employed to form a server farm as in Figure 3, together with a monitoring server, and a DNS server.

[2] One can test-drive this sample application by visiting the following Web site: http://duwamishonline.com

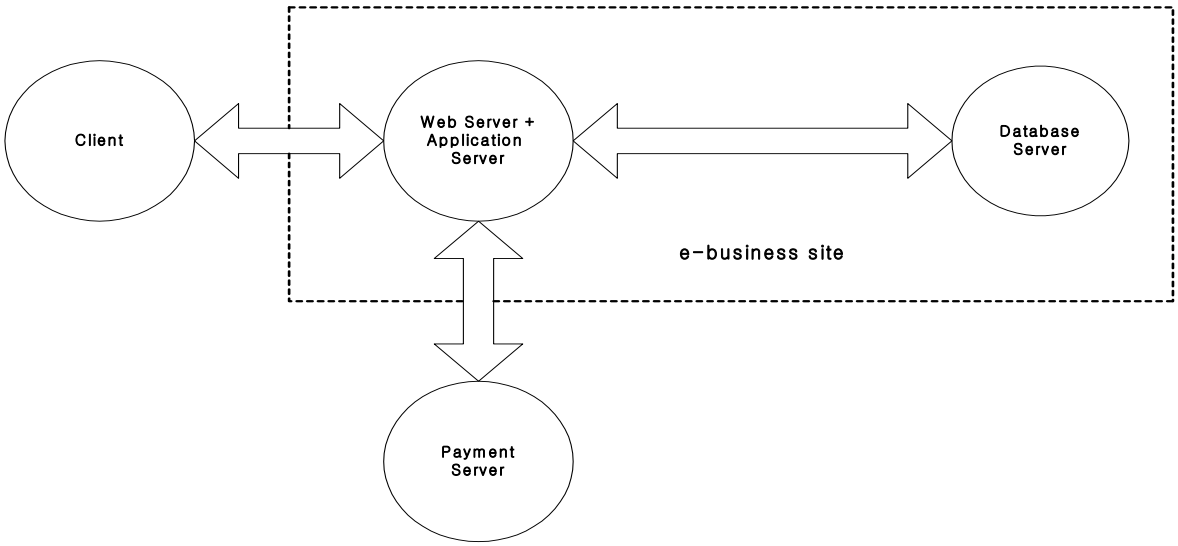

Figure 2: A physical two-tier architecture in its simpliest form

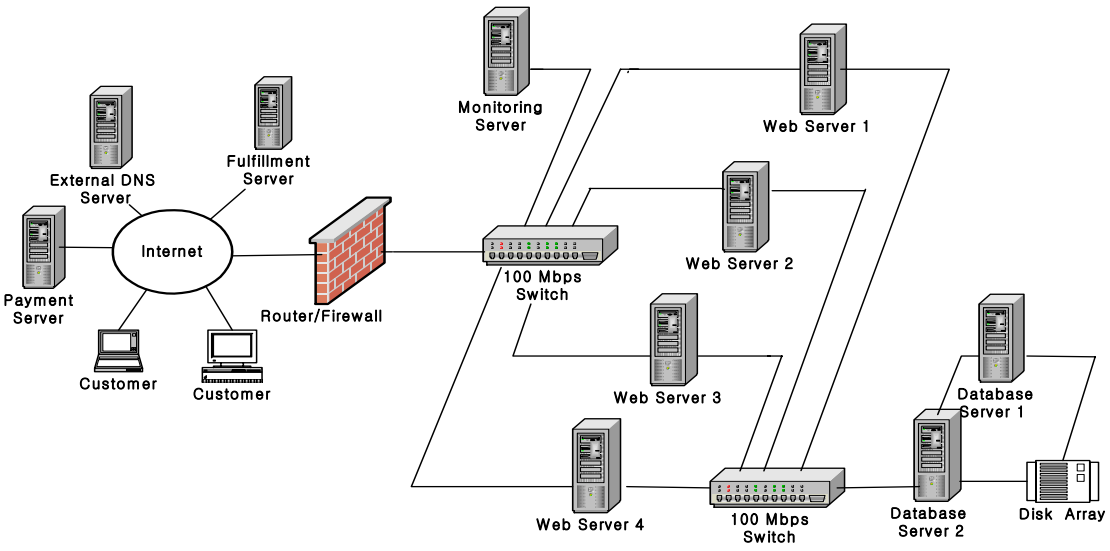

Figure 3: A physical two-tier architecture in reality

Another way to implement the logical architecture is a physical three-tier architecture where the business logic layer resides in application servers.

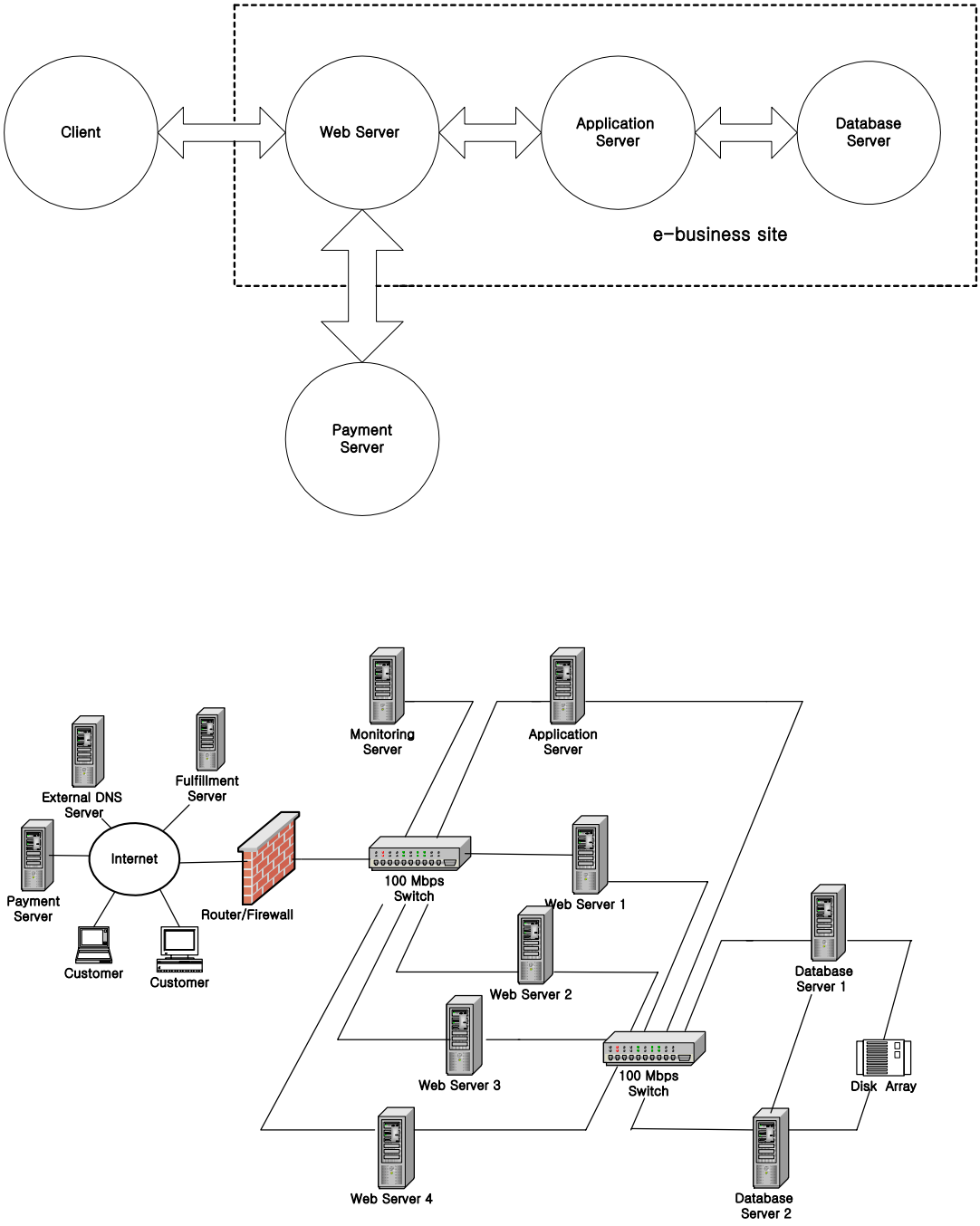

Figure 3: A physical three-tier architecture

To draw architecture diagrams shown above, we need a capacity planning.

3. Capacity planning and stress test

No e-commerce Web site wants to present its visitors with a poor user experience such as slow response times, timeouts and errors, and broken links. Web users are fickle, and when presented with a poor user experience they'll often turn to other sites to find what they're

looking for. To prevent this, site operators must build an infrastructure that can handle not only average levels of demand but also peak levels and beyond (de Klerk and Bender 2000). This is the rationale for a capacity planning.

When performed correctly, capacity planning makes it possible to calculate how much computing hardware is necessary to handle the demand that thousands of users can put on a Web site. These calculations can help site operators find "weak links" that cause performance degradation. By adding hardware or by redesigning dynamic pages or other CPU-intensive tools, site operators can resolve these weak links.

Site capacity is determined by the following three factors:

1. *Number of users.* As the site attracts more users, capacity must increase, or users will experience performance degradation.
2. *Server capacity and hardware configuration.* By upgrading the computing infrastructure, site operators can increase the site's capacity, thereby allowing more users or more complex content, or a combination of the two.
3. *Site content.* As the content of the site becomes more complex, it can result in the servers having to do more work per user, thereby lowering the capacity of the site. Conversely, site operators can sometimes increase capacity by simplifying content, eliminating some database use and dynamic content and serving simpler HTML pages.

Capacity planning can be expressed as a simple equation:

Number of supported users = Hardware capacity / Load on hardware per user

where "number of supported users" refers to concurrent users, and "hardware capacity" refers to both server and network capacity.

This equation suggests two corollaries:

1. Decreasing the load that each user puts on the hardware can increase the number of supported users.
2. Configuring the site infrastructure to increase hardware capacity can increase the number of supported users. Options include scaling the hardware *horizontally* (adding more servers) or *vertically* (upgrading the existing servers).

If we want to increase the complexity of the content of the site, thereby increasing the *load on hardware per user* and still maintain the *number of supported users*, then the hardware capacity must be increased. If we want to support more users, we must either simplify the

site content or increase hardware capacity.

Most e-commerce sites use some form of dynamic content that can be provided by a wide variety of Internet and database technology. At its simplest, dynamic content involves the Web server contacting a database, retrieving data, formatting it, and then sending it to a user's browser as a Web page. For example, if a user wants to see information on a specific product, the server might contact a SQL database to retrieve the product's description, a photo, price information, and whether or not the product is in stock. The resulting page would display in the user's browser using conventional HTML as if it were a static Web page, but it would be created on the fly by the server when the user requests it.

To perform a hardware capacity planning for a site with dynamic contents, the following factors have to be considered:

- CPU power (both Web and database servers)
- Memory utilization (both Web and database servers)
- Disk access speed (primarily database Server)
- Network utilization (LAN and Internet)

A trade-off emerges among the hardware resources since their cost implications are different. By conducting a what-if analysis using the data from a performance model and a cost model, alternative architecture could emerge. It is even possible to modify business, functional and customer behavior models based on the what-if analysis. Capacity planning can therefore be described as an iterative process as shown in Figure 4.

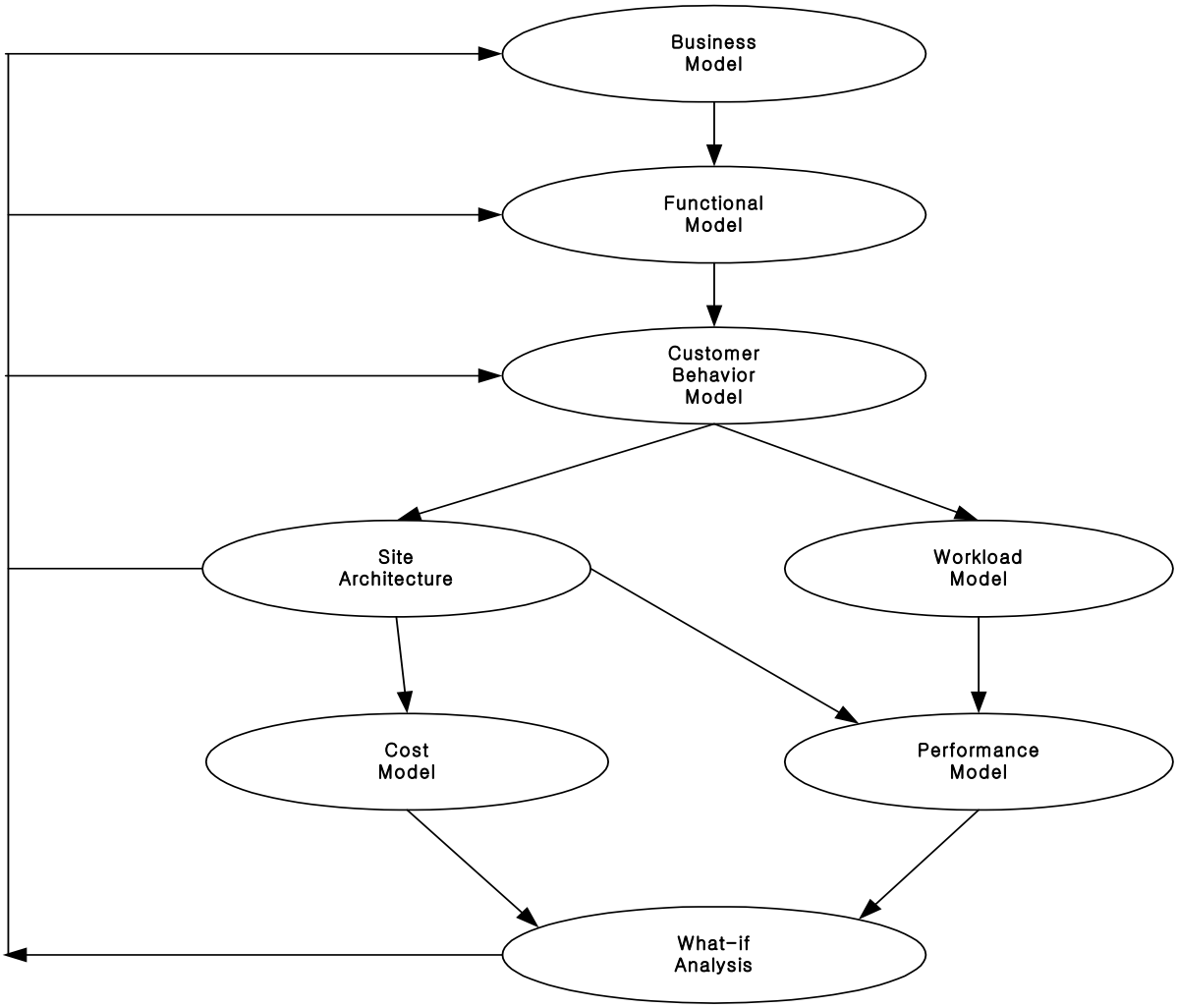

Figure 4. Iterative process of capacity planning

Once an e-commerce infrastructure has been built following a careful capacity planning, a stress test is needed to guarantee a successful operation of the Web site. This is so since software and hardware components are interweaved in a complicated manner in e-commerce applications: planning alone cannot eliminate hidden risks entirely. As its name suggests, it is a test to load a Web site with thousands of simultaneous access or hits, and see if the infrastructure supports the transaction volume with satisfactory response time. If a capacity planning is needed before building an e-commerce infrastructure, a stress test is needed after the infrastructure has been built but before going into a production mode. As shown in Figure 5, capacity planning and stress test compliment each other and we argue that a circular relationship exists between the two.

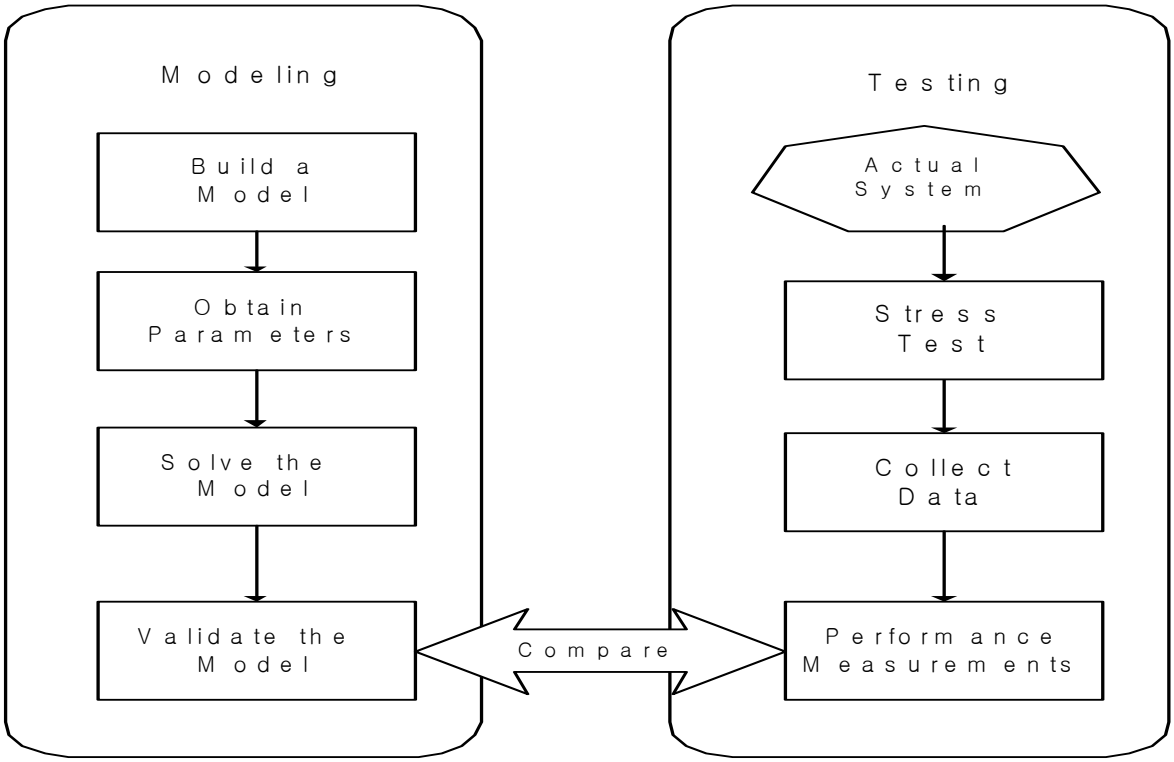

Figure 5. Capacity planning(modeling) and stress test

Since the purpose of this paper is to evaluate alternative physical architectures to implement the logical three-tier architecture, we have conducted a stress test only.

4. Research Design and Test Results

In order to conduct a performance test of a Web application, we need to coordinate with hundreds or even thousands of real users to access the Web site within a designated period of time. This is unrealistic and expensive. Another way is to work with a testing tool that can reproduce such user loads. Basically, a minimal number of client computers are used to stimulate a large number of virtual users, concurrently requesting predefined pages or *URLs* (Uniform Resource Locators) of the Web site. Each of these virtual users emulates the exact communication protocols between a real Web browser and the Web server (Ching et al 2001). Many such performance testing tools are available, and in this paper, we chose the Microsoft's Web Application Stress Tool (WAST) that can be downloaded free of charge from http://webtool.rte.microsoft.com/.

As for the Web application to be tested, we wanted to use Duwamish Online, a sample Web application developed by Microsoft engineers in accordance with the logical three- tier architecture shown in Figure 1. Source code including image files is available for a free

download[3]. Though modular in nature, the application turned out to be quite complicated for implementation, partly due to an extensive usage of XML/XSL to generate eletronic catalogs. Because of a time limitation, we chose another sample application that employs Active Server Pages with Visual Basic scripts to simulate e-commerce shopping mall (Lee et al. 2001). As a business layer, we added a discount policy module that utilizes transaction history records and a credit card verification process with an external system.

With WAST, a Visual Basic script is used on client computers to generate requests. Normally, the objective of a stress test is to load the Web server with as many requests as possible to determine a maximum throughput per second. Maximum throughput occurs when a sudden increase in operation latency or page request queue is observed..

Since our objective is to evaluate alternative physical architectures, we ran the test with two-server system first. Under the physical two-tier architecture, the Business Logic layer dealing with customer discounts (SQL stored procedures combined with Visual Basic scripts) and credit card verification were placed on the Web server. We started with two million page hits per day. Two million page hits per day is an average of 23.14 pages per second: (2,000,000 pages/day / (24 hr/day * 60 mins/hr * 60 sec/min).

However, since the page views don't all come at the same time, we introduced the 80/20 rules: 80% of the hits will be received in 20% of the time. This is the peak loading. Using the 80/20 rule, peak usage will be: (0.8 * (2,000,000))/(24 * 60 * 60 * 0.2) = 92.59 pages/sec while off-peak hits were: (0.2 * (2,000,000))/(24 * 60 * 60 * 0.8) = 5.78 pages/sec. Starting from a minimum of 5 pages per second, we increased the load by 10-pages interval up to 90 pages per second.

To apply WAST, specific pages on the Web server have to be designated. To simulate a typical user activity at the site, we defined the following usage patterns or activity mix and designated the Web pages accordingly:

- 50 percent Category Browsing
- 30 percent Item Details Browsing
- 20 percent Add to Shopping Cart
- 10 percent Checkout

After a warm-up period, we ran the scripts for twenty minutes and then took measurements.

[3] The following URL will open a Web page, from which download link is available: http://msdn.microsoft.com/voices/sampleapp12112000.asp

This allows the various caches (memory, disk, IIS, SQL, and ASP programs) to get a reasonably stable state. We collected the following two statistics: Time To Last Byte (TTLB), which shows how fast the results are presented to the client by the servers and Responses Per Second(RPS), which shows how many pages are sent out by the servers (Chang 2000).

Then we changed the hardware configuration. Now, under the physical three-tier architecture, the Business Logic layer was placed on a separate server called Application Server as in Figure 3 above and the stress test was conducted toward it.

After a test was run, test statistics generated by WAST, the Web Application Stress Test tool, were exported into an Excel spreadsheet. A series of Excel sheets were therefore created and were integrated into a single sheet. Figure 6 is based on the final, integrated test results. Here, WAS2 and WAS3 refer to 2 and 3 physical tiers, respectively, with the term WAS representing Web Application Stress (no particular meaning). The test results show that a physical two-tier architecture is superior to a physical three tier architecture.

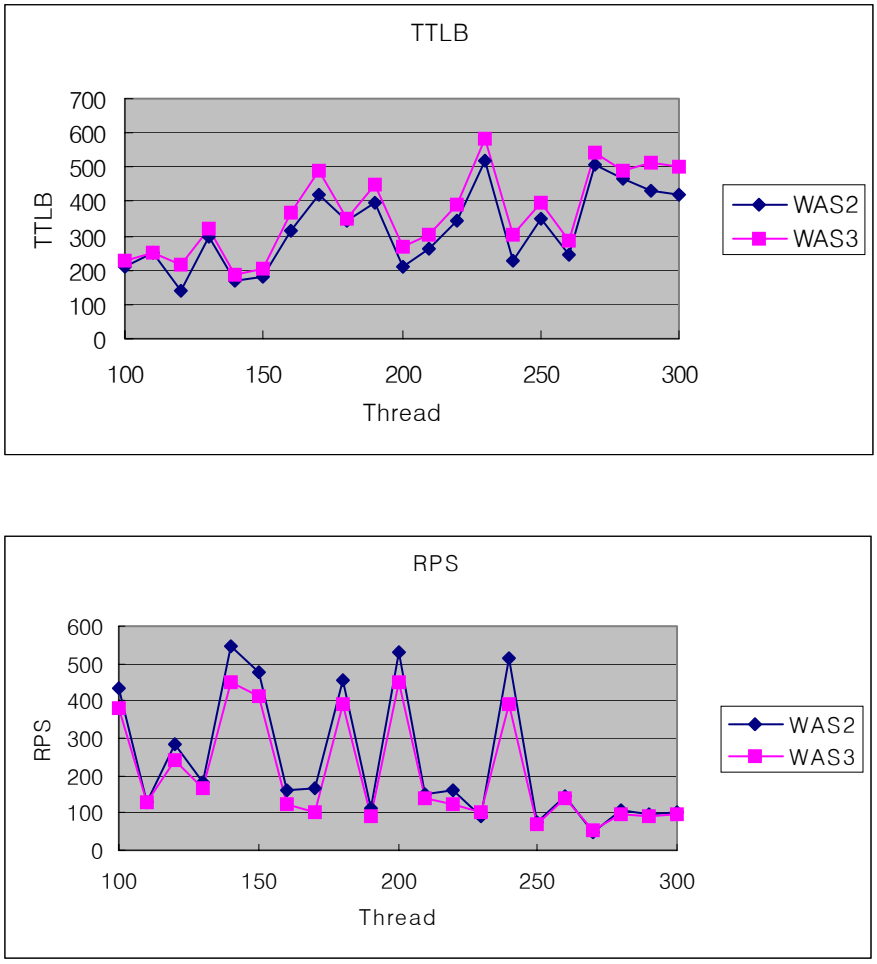

Figure 6. Test Results

A more rigorous statistical testing is in order. We therefore applied Wilcoxon Matched-Pairs Signed-Ranks Test to the above data, using SPSS for MS Windows. The first null hypothesis to be tested is that there is no difference in the paired ranks of TTLB(Time To Last Byte) between the two physical architectures. Listed below are the non-parametric test statistics. Here, TTLB3 and TTLB2 represent TTLB under 3-tier and TTLB under 2-tier architecture, respectively, with LT, GT, EQ implying Less Than, Greater Than, and Equal, respectively.

Mean Rank	Cases	
.00	0	- Ranks (TTLB3 LT TTLB2)
11.00	21	+ Ranks (TTLB3 GT TTLB2)
	0	Ties (TTLB3 EQ TTLB2)
	21	Total

Sum of Ranks = 231.00 Z = -4.0145 2-Tailed P = .0001

The sum of ranks far exceeds the upper critical limit of 167. At the significance level of less than 1 percent (p=.0001), we reject the null hypothesis of no difference, and adopt the alternative hypothesis that there indeed exists a difference in TTLB between the two physical architectures.

Similarly, we applied the Wilcoxon rank test to the RPS (Requests Per Second) data, and present the results as follows. In essence, RPS represents the number of Active Server Pages processed by IIS(Internet Information Services), the Web server daemon.

Mean Rank	Cases	
11.71	17	- Ranks (RPS3 LT RPS2)
3.67	3	+ Ranks (RPS3 GT RPS2)
	1	Ties (RPS3 EQ RPS2)
	21	Total

Sum of Ranks = 199.00 Z = -3.5093 2-Tailed P = .0004

Here again, RPS3 and RPS2 represent RPS under 3-, 2-tier architectures respectively, with LT, GT, and EQ defined as before: Less Than, Greater Than, and Equal. Here again, the null hypothesis of no difference is rejected at the probability level of less than 1 percent (p=.0004). The sum of ranks far exceeds the upper critical limit of 167.

5. Conclusion

Since most e-commerce sites are designed as a logical n-tier architecture, it can often be construed as requiring that many physical tiers. After running two sets of configuration tests on a shopping mall application, however, we verified that one to one correspondence does not need to exist between logical and physical architectures: physical two-tier platform performed better because it minimized cross-machine communication.

In this two-tier configuration, the Web servers run all of the ASP pages for the Web site and all of the COM+ components, and the second tier runs the database server. Though multiple Web servers are often deployed to form a server farm in practice, we employed a single Web server. Of course, separate servers were introduced to function as a database server and an application server, together with another server to act as a credit card company server. We do not think this will affect the conclusion. The tests we ran with business logic components running on the middle tier had lower throughput and slower response time.

This could have a far-reaching implication. It would imply that e-commerce sites do not need to spend their precious dollars on purchasing separate application servers. Instead, the various stored procedures dealing with business policy can best be placed on Web servers. Of course, there is a limitation to this line of reasoning, since we have not performed the stress test under a Web server farm environment. We plan to apply the stress test toward a real e-commerce site as a future research project.

An important objective of this paper was to show how to utilize a stress test to evaluate alternative architectures for an e-commerce site. Since the stress test tool is freely available for download, MIS professionals need to get acquainted with the tool. Time has come for the MIS professionals to embrace architecture evaluation as their core responsibility: gone are the days when Web sites could simply add servers to improve response time. Since the throughput, or quick response time, is determined not by the number of servers alone but by a combination of factors including software deployment, a stress test would emerge as a useful tool to evaluate alternative architectures.

Stress test alone is not sufficient: we need a thorough understanding on how the load is created by various software modules and handled by various hardware devices in e-commerce. Combined with a capacity planning, a Web application stress test could turn out to be a powerful tool to improve the performance of e-commerce sites in a cost-effective manner. Our future research would be directed to investigate the complimentary nature of the two.

REFERENCES

Bass, L. Clements, P. and Kazman, R. Software Architecture in Practice, Addison-Wesley 1997.

Booch, Rumbaugh, and Jacobson, The UML Modeling Language User Guide, Addison-Wesley, 1999.

Chang, C. Web Application Stress Test And Data Analysis, *Unisys Consulting Services,* McLean, Virginia, January 20, 2000. Available for free download from the following site: http://webtool.rte.microsoft.com.

Ching, A. Silva, P. and Wagner, A. Performance Testing with the Web Application Stress Tool, Microsoft Developer Network, January 2001

de Klerk, L. and Bender, J. Capacity Planning, Microsoft Enterprise Services White Paper, *April 2000.*

Kazman, R. Klein, M. and Clements, P.: ATAM: Method for Architecture Evaluation, Carnegi Mellon University, Software Engineering Institute, August 2000.

Lee, S.B., Park, K.R., and Lee, K.M. Active Server Pages Utilization to Build E-Commerce Shopping Mall(Korean), Yunhaksa Publishing Co., Seoul, Korea, 2001.

Lee, Y.H. A Study on the Effectiveness of an Electronic Commerce Model, Proceedings of INFORMS-KORMS Seoul 2000 Conference, June 2000, Seoul, Korea.

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

Reichheld, F.F. and Schefter, P. E-Loyalty: Your Secret Weapon on the Web, Harvard Business Review, July-August 2000.

Silva, P. and Edwards, E.D. Introduction to the Duwamish Online Sample Application, Microsoft Developer Network, July 2000. Available from the following URL: http://msdn.microsoft.com/library/default.asp?URL=/library/techart/d5dplywindna.htm