# A Step towards a Benchmark Repository for E-commerce

Peter Bodorik[1], Dawn Jutla[2], Lihong Cao[1] and Yie Wang[1]
[1]Faculty of Computer Science, Daltech, Dalhousie University
[2]Faculty of Commerce, Saint Mary's University
dawn.jutla@stmarys.ca, bodorik@cs.dal.ca

## Abstract

*A benchmark is used to measure and compare the performance of like systems. It is also used to estimate the scalability of a system in terms of the number of users and/or transactions that a system can support and system response times under different loads and hardware/software deployment platforms. Because of the multidimensional aspects of e-commerce and the various emerging and distinct e-commerce business models, one single benchmark is not suitable for the purposes of examining scalability and determining throughput and response time of an e-commerce application under consideration. The diverse needs of small-to-medium enterprises (SMEs) and big business provide additional motivation for a benchmark suite for e-commerce.*

*This paper advocates development of a suite of benchmarks of distinguished business models that represent existing and emerging e-commerce applications in terms of their distinct features. In particular, the paper develops a benchmark for one such business model, a model that is based on a cybermediary or electronic broker, e-broker. Its distinguishing feature is the buy transaction consisting of a set of nested, distributed transactions that perform credit checks, user authentication, and product availability searches. Two implementations of the benchmark specification are described: one based on Microsoft's COM technology while the other one is based on CORBA technology. Sample benchmark results are also presented and discussed in order to demonstrate the usefulness of the benchmark when examining usage of distinct technologies in application implementation and also when examining the scalability of the application.*

## 1. Introduction

Benchmarking may not be viewed as exciting as development of new products or tackling problems in order to find solutions that lead to new technologies or products. Nevertheless, benchmarking is important as it is used to measure and compare the performance of like systems under controlled conditions, and for fine-tuning systems' performance. Equally importantly, it is also used to determine the characteristic performance of a system under various scenarios. Typical use of a benchmark is in determining the scalability of a system in terms of the number of users and/or transactions that a system can support and system response times under different loads and hardware/software deployment platforms.

E-commerce exhibits multidimensional aspects and operates in various environments and, consequently, different e-commerce business models have been developed. The models exhibit distinctive characteristics in terms of the user-server interaction, the number of DBs accessed, transactions' characteristics, and the business rules employed. One single benchmark is thus unsuitable for the purposes of examining scalability and determining throughput and response time of diverse e-commerce applications. The diverse requirements of small-to-medium enterprises (SMEs) and big business additionally motivate the need for a benchmark suite for e-commerce.

This paper advocates development of a suite of benchmarks of distinguished business models that represent existing and emerging e-commerce applications in terms of their distinct features. In particular, the paper develops a benchmark for one such business model, a model that is based on a cybermediary or electronic broker, e-broker. Its distinguishing feature is the buy transaction consisting of a set of nested, distributed transactions that perform credit checks, user authentication, and product availability searches. Two implementations of the benchmark specification are described: one based on Microsoft's COM technology while the other one is based on CORBA technology. Sample benchmark results are also presented and discussed in order to demonstrate the usefulness of the benchmark when examining the effect of distinct technologies on performance and also when examining the scalability of the application.

The paper is organized as follows. Section 2 briefly describes the e-broker business model. Section 3 describes the benchmark design including graphical layout, transactions, databases, and security. Section 4 briefly outlines the two implementations while Section 5 presents and compares the benchmarking results. Related work is detailed in section 6 while the final section provides a summary and concluding remarks.

# 2. Electronic Commerce E-Broker Model

The Internet e-commerce business models impact the e-commerce benchmark applications in terms of distribution types and makeup of the business transactions, frequency of invocation of individual transactions, and the number of accesses to remote databases over the Internet. Various definitions of business models for e-commerce include " an approach a company takes to make money" and "entire collection of activities that support commercial activities on a network." Three distinct classes of e-commerce business models were described in [Jutla 1999a]: e-broker, manufacturer and auction models. Each model represents a category of existing e-commerce applications. This section briefly describes one category represented by the e-broker model. It also very briefly describes the other two models, the auction model and the manufacturer model. It should be noted that the categories that these models represent are not exhaustive in that there are existing and emerging e-commerce applications that do not "fit" either of the three models and that further models need to be developed.

## 2.1 E-broker Model

The e-broker model is characterized by an intermediary between suppliers of goods and/or services and the customer. The intermediary, or e-broker, adds value to its on-line supplier sites, either by marketing a large range of similar-type products from one site, or through enabling comparison shopping, or through facilitating coalition industries such as real estate and the automotive industry that provide multiple company listings. Other ways of adding value is through creation of community, facilitation of customization, and enhanced customer care through user services such as personal account information. Negotiation facilities also provide added value on an online site. The e-broker model can support the sourcing of a product or service from many suppliers and may provide a customer with a choice of products/services, or the best price/delivery time for a product/service, or bulk discounts.

Figure 2.1 shows the e-broker model described in terms of general interactions among the business entities. This model incorporates access to databases internal to the company and to external databases of some other companies. Security and payment system(s) are implicitly incorporated in the electronic commerce application itself. Normally, a limited stock is held by the e-broker company. Delivery can be made either from the e-broker or from the supplier. Some suppliers are not set up to deliver goods of one unit size but can only accommodate delivery of multiple units; thus, in some instances, the e-broker may need to mediate delivery.

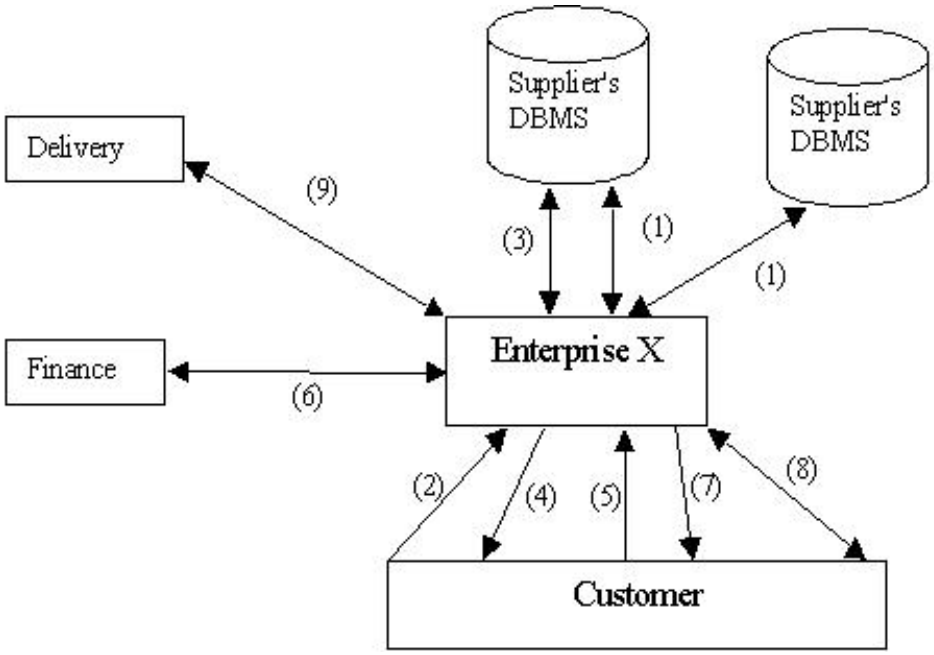


Figure 2.1 The E-broker Business Model

Consider Figure 2.1 in terms of showing how the electronic broker model works. In (1), Enterprise X (the broker) electronically sources quantities of BuyAlot (the product or service) from multiple suppliers or other e-brokers over the Internet so as to offer the best price and delivery terms. Thus, first interactions are those between the e-broker and the

suppliers. In (2), the customer makes the initial contact with the broker and proceeded with further interaction (navigation of pages) to obtain information about BuyAlot. This results in accessing a supplier's DB (3) and presenting the information to the customer (4). After obtaining the desired information about BuyAlot, the customer decides to purchase it and clicks on the buy button resulting in a request for purchase (5). This results in a query for credit card information. The information is sent to a third party, in this case a bank, which responds with an authorization number (6). Finally, Enterprise X sends the customer a notification that the order is confirmed, either on the spot or through e-mail (7). Order tracking service (8) allows the customer to check at what stage of delivery (9) the product is at present time.

The business model consists of querying multiple online supplier databases, requesting services from third party businesses and responding to the consumer. In a brokerage a function may be applied to select the supplier with the least cost, shortest delivery time or sufficient stock. In addition to the online consultation of the suppliers' databases, credit verification at a third party financial-service company (e.g. bank) is required.

Examples of simplified e-broker models include amazon.com and abe.com.

## 2.2 Manufacturer and Auction Models

Dell, Cisco and General Electric are example successes of the manufacturer business model. The manufacturer model implies that marketing and distribution are part of the company's operations. Note that here manufacturer encompasses the re-packagers as well. The virtual supply chain and forecasts for product demand support production planning. Unlike the e-broker model where the finished good is bought from suppliers and resold, the manufacturers create value-added products through their internal manufacturing processes. This model works best for organizations with configurable products, mature marketing staff and sophisticated customer service processes. Established businesses such as car and computer product manufacturers use this model.

The auction model, sometimes referred to as the Internet Exchange model, can be found in companies such as priceline.com and eBay.com. Basically this model works like a stock auction market where the buyer sets the price of the product through submission of bids and the willingness of suppliers to sell at the bidding price. Many auction sites charge a small fee to the suppliers when a sale is made and also may charge a fee for the listing of goods for sale at their site. Businesses that use this model require large customer bases in order to make money because of low cost services.

It should be stressed again that the business models discussed above do not represent all e-commerce applications and further models need to be developed.

# 3. WebEC Benchmark for the E-broker Model

The benchmark is targeted to the small-to-medium enterprises (SMEs). Business transaction definitions (browse, user registration, buy, shopping cart and order status), except for the buy transaction that will be discussed shortly, provided in this section are generic to any e-broker based e-commerce site. The benchmark design includes page navigation, a set of internal and external relations/tables, and transactions. They are briefly described in the following subsections. For further details interested readers should consult [Cao, 1999].

## 3.1 Navigation Design

The set of web pages that users navigate through includes: Welcome Page, Category Page, Products by Category Page, Product Page, Shopping Cart Contents Page, User Registration Page, Credit Card Info Page, Order Status Page, Order Confirmed and Thank-you Page. They are shown in Figure 3.1. With the exception of the pages for user registration and credit card information, pages have search and browsing to other pages as indicated by arrows. If the user can invoke a transaction(s) on a page, the transaction's name is also shown. The frequency with which pages and transactions are invoked will be discussed in a subsequent section.

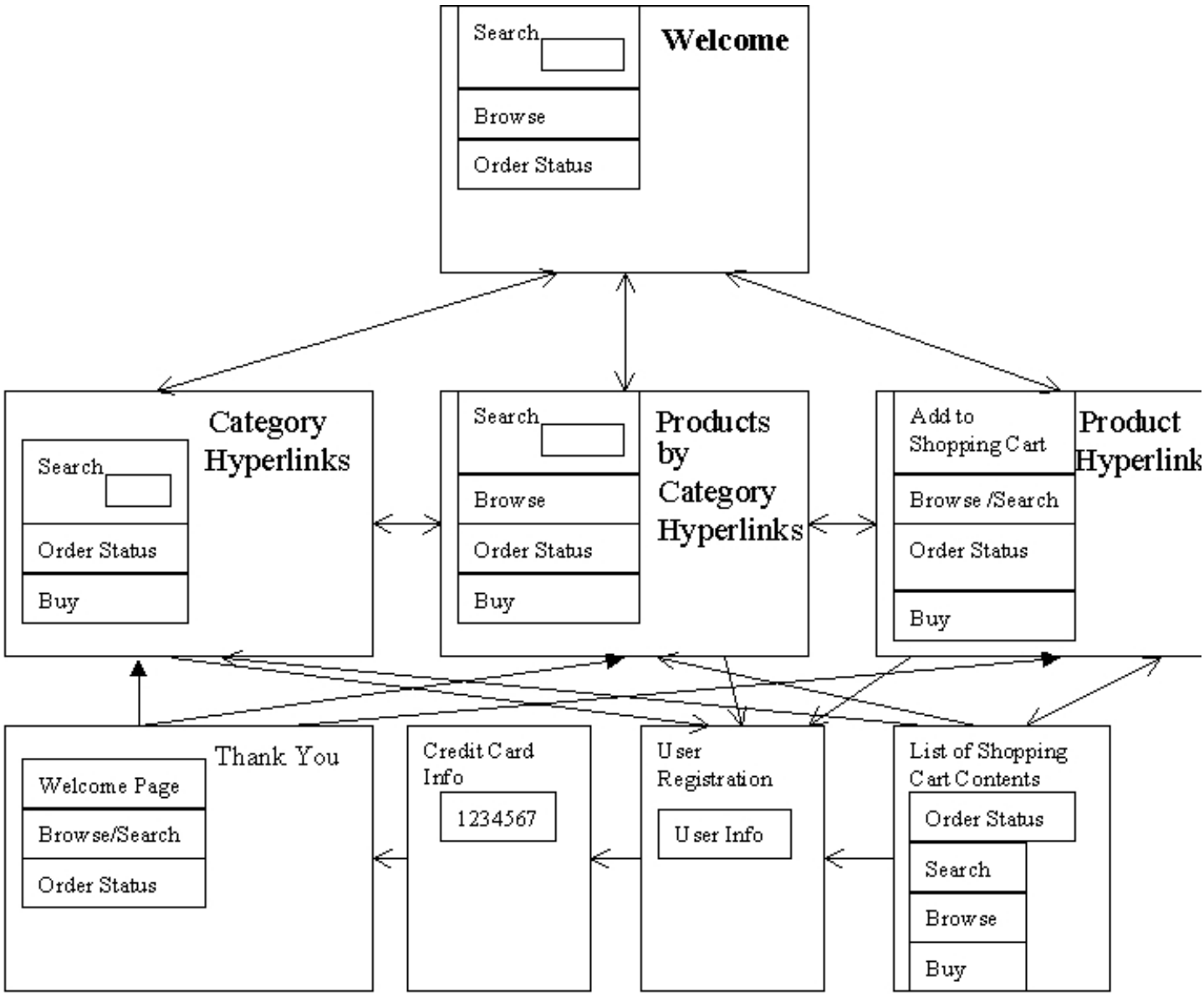


Figure 3.1 Navigation Diagram

## 3.2 Database Entities and Relationships

DBs accessed by the e-broker are either local or remote. The local database consists of twelve individual tables shown in Figure 3.2(a). The relationships, one-to-one and one-to-many, are also shown. The e-broker accesses tables of a remote DB of a finance company and tables of remote DBs of four suppliers. Four partner suppliers hold trusted relationship with the e-broker. The tables accessed in the bank's DB are shown in Figure 3.2(b) while tables accessed in the supplier DBs are shown in Figure 3.2(c).

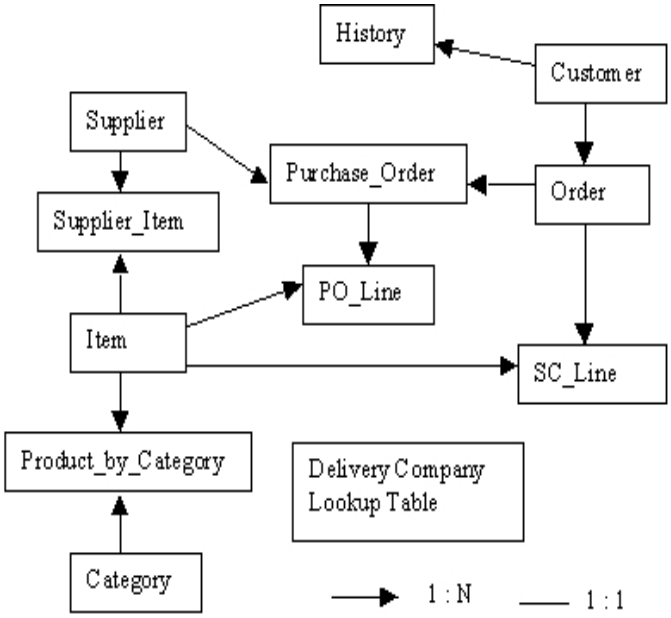


Figure 3.2 (a) Local Tables and Relationships


Figure 3.2 (b) Tables in a Remote Supplier's Database

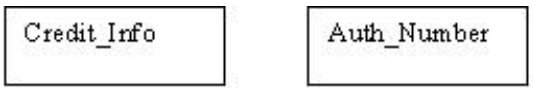


Figure 3.2 (c) Tables in a Finance Company's Database

**Figure 3.2** Local and Remote Tables

In Figure 3.2, SC_Line stands for a shopping cart line and PO_Line for the purchase order line. The authorization number in Figure 3.2(c) is used to keep track of the most updated number assigned to customers from the credit card company.

## 3.3 Business Transactions

The WebEC benchmark specifies a web e-commerce transaction processing (OLTP) workload. It is a mixture of online transactions that simulate the activities found in the e-broker model environments. Similar to other benchmarks, the purpose of the WebEC benchmark is to retain the essential characteristics for a given environment or context, namely, the level of system utilization and the complexity of the operations [TPC 1997].

In general, transactions may be simple in that they are triggered by the customer and they result in a simple request-reply interaction between the customer/browser and the web-server. (In the context of DBs, such an interaction does not constitute a transaction at all as there is not DB access.) Example is the Welcome Page transaction. Some transactions are simple and also cause access to a single either local or remote DB. Examples are Stock Level Update and Order Status transactions. There is one relatively complex transaction, the nested Buy transaction. It consists of a number of sub-transactions.

Transactions can also be classified by whether they are triggered explicitly by the customer or whether they are sub-transactions triggered internally. Finally, when a transaction accesses a DB, it can perform either read-only operations or read/write operations. The classification of the transactions is shown in Table 3.1.

**Table 3.1** Summary of WebEC Business Transactions

| **Transaction** | **Triggered by customer/internally** | **ReadDB/WriteDB** | **Simple/Nested** |
|---|---|---|---|
| Shopping Cart Transaction | Customer | R/W | Simple |

| | | | |
|---|---|---|---|
| Customer Registration Transaction | Customer | R/W | Simple |
| Buy Transaction | Customer | R/W | Nested |
| Order Status Transaction | Customer | R | Simple |
| Welcome Page Browsing Transaction | Customer | - | Simple |
| Category Page Browsing Transaction | Customer | R | Simple |
| Product by Category Page Browsing Transaction | Customer | R | Simple |
| Product Page Browsing Transaction | Customer | R | Simple |
| Stock Level Update Transaction | Customer | R/W | Simple |
| Credit Verification Transaction | Internally | R/W | Simple |
| Online Supplier Contact Transaction | Internally | R/W | Simple |
| Delivery of Purchase Order Transaction | Internally | R/W | Simple |
| Customer History Update Transaction | Internally | R/W | Simple |
| E-broker Transaction | Internally | R/W | Simple |
| Order Confirmation Transaction | Internally | R | Simple |

When a transaction accesses a remote DB, it is a distributed transaction spanning different remote resource managers, e.g., remote database management systems (DBMSs) residing on the partner suppliers. As stated before, it is assumed that the partner suppliers have trusted relationships with the e-broker. This implies that they permit the use of X/Open XA interface for the purposes of distributed resource management and use of 2-Phase Commit. Many products support the XA interface including Oracle, Sybase, and Informix relational databases. It should be noted that such trusted relationships actually do appear in the e-commerce applications and have emerged as one of the features of e-commerce [Riggins 1998].

*The Buy Transaction*

This transaction is the most complex one, consisting of a number of sub-transactions, and is described here in a little more detail. It is a heavyweight read-write transaction that triggers several sub-transactions, namely the registration transaction (with a certain probability), the credit verification check, the online supplier contact, the internal delivery, the customer history update, and the order confirmation. Paying by credit card is chosen as the payment method and the Secure Socket Layer (SSL) is used as the security protocol. The buy transaction only executes if the user has something in the shopping cart. Figure 3.3 shows an example scenario for the buy transaction - different scenarios depend on the number of items in the shopping cart. In this particular scenario, the customer places one item in his shopping cart and triggers the buy transaction. Remote supplier number 2 is chosen to provide this item because it offers the best price or the delivery time. The specification of the buy transaction is as follows:

Input Generation:

- The random customer number (C_ID) is selected from IDs in the CUSTOMER table with C_ORDER_FLAG = “YA” (this customer has previously done a shopping cart transaction).
- The non-uniform random Credit Card number is selected using the NURAND (2047,1,10000) function.
- Tne random O_ID is selected among the O_IDs matching the C_ID in the ORDER table with the O_PO_FLAG = “NO”.
- A fixed 0.1% of the buy transactions are chosen at random to simulate failure and exercise the performance of rolling back update transaction. This is implemented by generating a random number rbk (rolling back) within [1,1000].

Transaction Profile:

- A web transaction is started.
- If a rbk > 1, there is no system failure; if a rbk =1, the failure occurs and the transaction is aborted.
- Credit Verification (sub)ransaction is started and committed.

- Online Supplier Contact (sub)transaction is executed.
- Delivery of Purchase Order (sub)transaction is executed.
- Customer History Update (sub)transaction is executed.
- Order Confirmation (sub)transaction is executed.
- The transaction is committed.

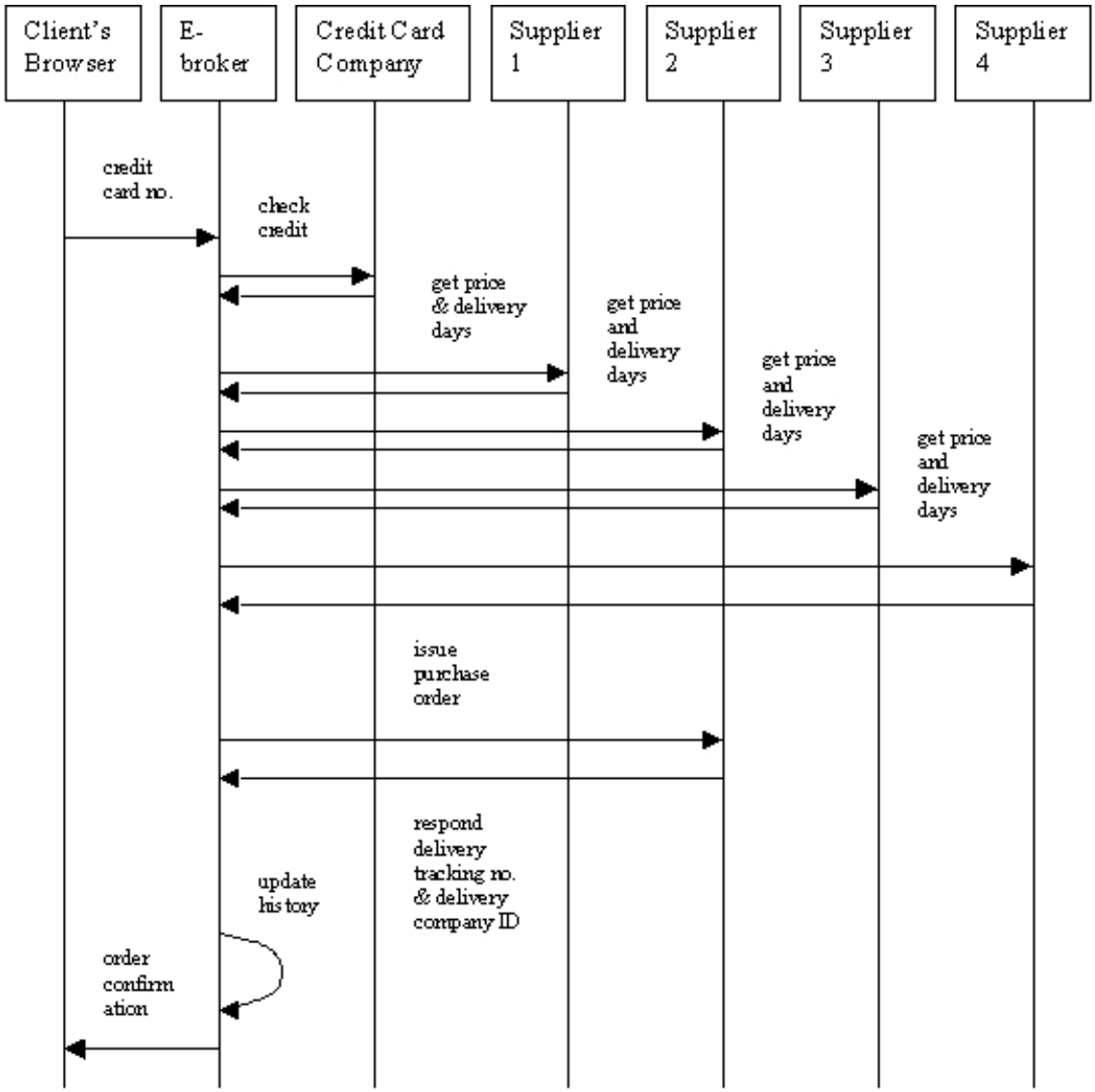


**Figure 3.3** One Scenario of the Buy Transaction

# 4. WebEC Benchmark Implementations

This section briefly describes two middleware implementations of the benchmark, one using Microsoft's COM technology while the other using Iona's CORBA technology. Both implementations are based on the three-tier client/server model and they also use the same Java transaction generator. The transaction generator, the three-tier client/server model, and the two implementations are described in this section. For further details on the COM and CORBA implementations see [Wang 1998] and [Cao 1999], respectively.

## 4.1 Transaction Generator

The transaction generator is written in Java to manage the running of the WebEC benchmark. The generator decides navigation of pages and which transactions are to be executed.

The transactions are generated based on the percentages of a mix of transactions that is specified in the input workload. A deck card algorithm is used to generate the appropriate number of transactions of each type. One or more cards in a deck are associated to each transaction type. The required mix is achieved by selecting each new transaction uniformly at random from a deck whose content guarantees the required transaction mix.

The generator governs the execution of the client side of transactions - pages are navigated through the browser while the generator waits "key time" periods to simulate user input. After a response is received from the server and the corresponding page is displayed, the generator waits a "think time" period to simulate the user's examination of the page and decision making on the next action (navigation, input, or transaction).

## 4.2 Three-tiered Client and Server Model

The conventional three-tiered client/server model, as given in [Jennings 1997], is used in both implementations. The subsystems that make up a three-tiered solution are data services, business services, and user services. Figure 4.1 shows the traditional three-tiered model.

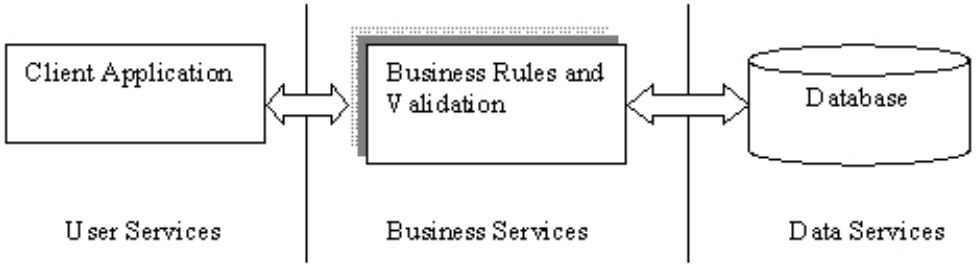


**Figure 4.1** Traditional Three-tiered Model

*Data Service Subsystem*

The data service subsystem in the three-tiered client/server model encapsulates functions, modules, and components that manipulate persistent data. The modules usually provide stored procedures, triggers, or external data access routines. On the Windows NT platform, Microsoft SQL Server and Oracle are examples of Relational Database Management Systems (RDBMS). The RDBMS can deploy data services to multiple users. Open Database Connectivity (ODBC) is the typical approach to managing most data access, which enables multiple users to obtain services from a specific Database Management System (DBMS). It is also the approach used here.

Note that the licensing agreement with the DBMS vendor prevents us to disclose which DBMS products were used.

*Business Service Subsystem*

This subsystem consists of the business rules that govern the behavior of the system. The implementation of the business logic in this tier avoids placing these rules in either the client or database server. This approach creates a highly scalable and robust system because the separation of logic from the client or database server reduces the resource requirement on them and eliminates the necessity to modify them if the business logic is changed.

The business logic in this subsystem includes source code to use the data service subsystem and performs operations on its data. For the WebEC benchmark system, it includes the business transactions that are applied on the data. The business services in this subsystem are also responsible for providing infrastructure and middleware including management of distributed transactions that is achieved by Microsoft's Transaction Server in the COM implementation of the benchmark and by OrbixOTS in the CORBA implementation.

*User Service Subsystem*

This subsystem is responsible for the user interface. Before the data reaches this subsystem and thus the user, the business logic has already been applied to the data by the business services (middle tier). In the WebEC benchmark system, the client

accesses the middle tier through the browser requests.

## 4.3 WebEC Benchmark Implementation Using Microsoft's COM Technology

This section briefly describes an example implementation of the WebEC benchmark using Microsoft's Component Object Model (COM) / Microsoft Transaction Server 2.0 (MTS) / Internet Information Server 4.0 (IIS) / ActiveX Server Page 2.0 (ASP) platform. The client accesses the Web server through the Internet. The client application is implemented using Active Server Page technology that adopts server-side scripting to dynamically create HTML responses. The clients' application is written using the ASP script language. IIS serves as the web server that accepts ASP requests from the ASP page sent from the client's browser.

The business services in this subsystem are responsible for providing infrastructure and middleware. The business logic of the WebEC benchmark is implemented as COM components. MTS groups the COM components under transactional control. As middleware, the MTS is used to coordinate and monitor the transactional state of the COM components running under its control as these components interact with various transactional systems, such as remote SQL Server or Oracle databases. MTS is tightly integrated with IIS and the ASP engine.

When business logic code in one COM component needs to access a remote supplier's database, the database connectivity technology, ActiveX Data Object (ADO), is used to interface with relational databases through Open Database Connectivity (ODBC).

Figure 4.2 shows the specific configuration of the three-tiered model. The business service subsystem consists of Microsoft Internet Information Server (IIS), Active Server Page (ASP) engine, Component Object Model (COM) elements, and Microsoft Transaction Server (MTS). Although only two DBs are shown they represent more then two. The figure shows that the RDBMSs in the Data Service tier are connected to the Business Services tier via a network. This represents a network connection to either a local DB via, perhaps, a LAN, or a connection to a remote DB, in which case the network represents a WAN or Internet.

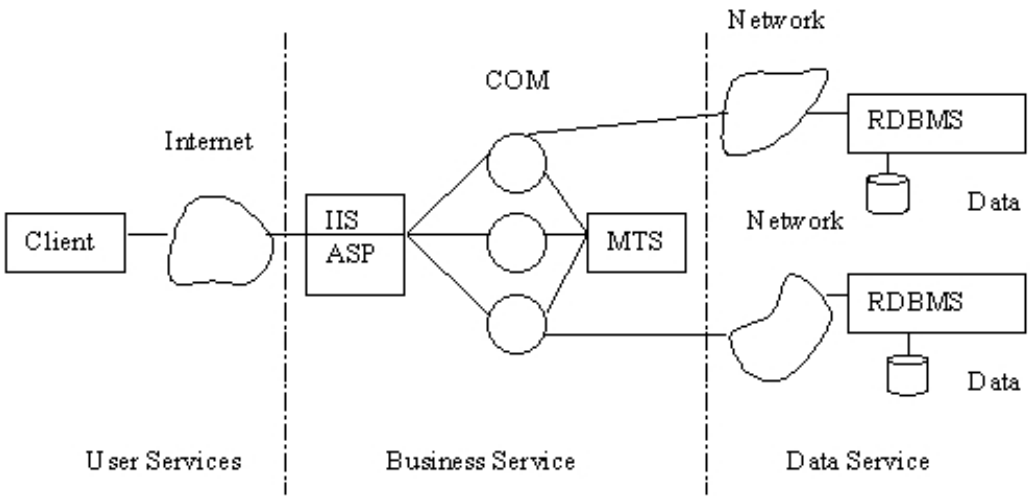


**Figure 4.2** Three-tier Model using COM

Along with the IIS and ASP servers that reside on the middle tier in the three-tiered client and server model, we include the Microsoft Transaction Server (MTS) as transactional middleware. MTS groups and manages the COM objects that consist of the business logic, i.e., nine business transactions, in the WebEC benchmark.

As mentioned above, MTS coordinates and monitors the transactional state of COM components running under its control as they interact with various transaction systems, such as an SQL Server or Oracle database. For example, in the COM component, buy.contact, suppose item 1 is selected from remote supplier 1, and item 2 is selected from remote supplier 4. The quantity of item 1 in supplier 1 will be updated by the deduction of the quantity the customer ordered. Similarly, the quantity of item 2 in supplier 4 is updated. These two update operations in this COM component are treated in one atomic transaction by the MTS. If any of them fails, these two operations will be undone. We set the attribute of each COM component in the WebEC benchmark to the option of "Require a Transaction". Thus all transactions, including the nested transactions, are managed by MTS.

## 4.4 WebEC Benchmark Using CORBA Technology

This section briefly describes an example implementation of the WebEC benchmark using a IONA's CORBA technology.

In the 3-tier client/server architecture, the client is a Java applet downloaded by a Web browser. The middleware is the OrbixOTS server and business components. These business components are shopping cart component, customer registration component, buy component, order status component, category browsing component, product by category browsing component, product browsing component, and stock level component. All business components are implemented as C++ classes. The third tier consists of RDBMSs.

Figure 4.3 illustrates the typical workflow of the WebEC benchmark. It demonstrates how the Web-based client interacts with its server:

1. The client makes the initial requested for a page.
2. Web browser receives and downloads an HTML page that includes references to embedded Java applets.
3. Web browser requests the Java applet from the HTTP server.
4. The HTTP server retrieves the applet and downloads it to the browser in the form of bytecodes. Web browser loads applet into memory.
5. Applet invokes CORBA server objects. The Java applet includes the IDL-generated client stub classes, which let it invoke objects on the ORB server.
6. Server objects access the database according to the defined business logic.
7. Server objects return the results back to the Web client, and the results are displayed on the client's browser.

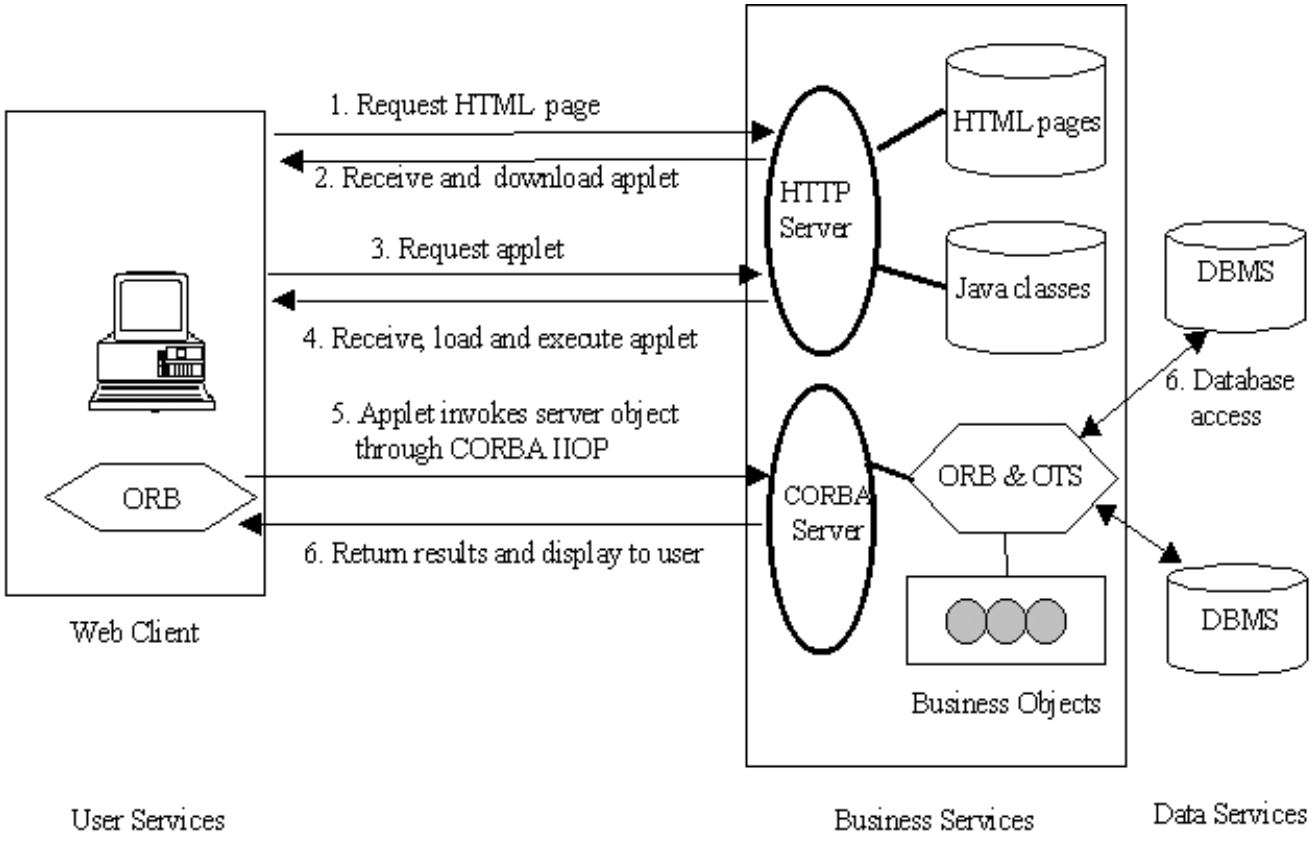


**Figure 4.4** The Workflow of WebEC Benchmark

# 5. Experiments

The purpose of the experiments is to examine the usefulness of the benchmark when considering deployment technology of an E-commerce application and when examining the issue of scalability. This section is provided to show example use and output of the WebEC benchmark only. The experiments are performed in a university lab with the available low-end equipment. In an SME business setting, the benchmark would exercise more powerful servers (and possibly a larger number of servers) with

a much wider range of users, in the thousands or tens of thousands of hits.

## 5.1 Performance Metrics

The performance metrics, response time and transaction throughput, are measured by running the WebEC benchmark application while varying a number of parameters such as the "think time", the "key time", the input transaction workload and the number of clients. During the measurement, the system log for the benchmark system is maintained.

The mix of transactions in the WebEC benchmark system represents a complete business cycle. It consists of multiple business transactions namely the Shopping Cart Transaction, the Buy Transaction, the Registration Transaction, the Order Status Transaction, the Welcome Page Browsing Transaction, the Category Page Browsing Transaction, the Products by Category Page Browsing/Searching Transaction, the Product Page Browsing/Searching Transaction, and the Stock Level Transaction.

The transaction throughput is the total number of completed business transactions divided by the elapsed time of the measurement interval. That is, the throughput is a number of business transactions processed per minute. If a transaction is aborted, say due to a deadlock, it is restarted by the transaction manager. The throughput of the WebEC benchmark system depends on both the system response and human interaction such as clients' keying and thinking time and number of users, and percentage load by transaction type.

The response time is the total individual response time of each business transaction divided by the number of transactions during the measurement interval. The response time is system oriented and it excludes the human interaction times, i.e., the "key time" and the "think time". From the view of execution time, a WebEC transaction starts when the request for a web page is triggered, and ends when the committed or aborted result is completely loaded on to the browser's screen.

## 5.2 Statistical Model

Over the measurement interval of the WebEC benchmark application, we need to maintain a percentage of mix for each transaction type as illustrated in Table 5.1. The first data set supports a ratio of 32% order related transactions (Shopping Cart, Buy, User Registration and Stock Update) and 68% browse type transactions (Welcome Page, Category Page, Products by Category, Product Page, Search and Order Status). The second data set represents 22% order related and 78% browse type transactions. Transaction types are selected at random while maintaining the required percentage of mix for each transaction type over the measurement interval. Table 5.2 illustrates the think and keys times used in the implementation.

**Table 5.1 Percentages of Mixed Transactions**

| Business Transaction Types | 1st Set | 2nd Set |
|---|---|---|
| Shopping Cart Transaction | 11% | 10% |
| Buy Transaction | 11% | 6% |
| Registration Transaction | 9% | 4% |
| Order Status Transaction | 10% | 4% |
| Welcome Page Browsing Transaction | 10% | 8% |
| Category Page Browsing Transaction | 12% | 18% |
| Products by Category Page Browsing/Searching Transaction | 12% | 22% |
| Product Page Browsing/Searching Transaction | 24% | 26% |
| Stock Level Update Transaction | 1% | 2% |

**Table 5.2 Keying Times and Thinking Times**

| Transaction Type | Key Time(sec) | Think Time(sec) |
|---|---|---|
| Shopping Cart | 0 | 1.5 |
| Buy | 1 | 2 |

| | | |
|---|---|---|
| Registration | 3 | 1 |
| Order Status | 1 | 1.5 |
| Welcome Page Browsing | 0 | 0.5 |
| Category Page Browsing | 0 | 2 |
| Products by Category Page Browsing/Searching | 0 | 1.5 |
| Product Page Browsing/Searching | 0 | 1 |
| Stock Level Updating | 0 | 1.5 |

## 5.3 Test Plans and Results

It should be noted again that the results and analysis are given as examples on the use of a benchmark in analyzing system performance.

Four test plans are designed to measure the performance of the WebEC benchmark application by varying the WebEC benchmark parameters. The parameters varied are the think and key time, transactional workload percentages, and the number of concurrent clients using the WebEC system.

The parameters that are varied give only examples of how the benchmark can be used to examine the system performance, here defined as system throughput and response time. Different key and think times are used to examine the effect of the user behavior (fast typists and thinkers versus twice as slower typists and thinkers) on the system performance. Transactional workload (see Table 5.1) examines the effect of different mix of transactions on the system performance: the 1st set has almost twice the number of buy transaction than the 2nd one (11% vs. 6%). Finally, the number of concurrent clients examines scalability of the system.

The four test plans are as follows:

1. Use the 1st set of the workload characteristics in Table 5.1 and the original set of the think and key time in Table 5.2. Obtain the response time and transaction throughput when there are 5, 10, 15, 20, 25 clients, respectively.
2. Use the 2nd set of the workload characteristics in Table 5.1 and the original set of the think and key times (as shown in Table 5.2). Obtain the response time and transaction throughput for 5, 10, 15, 20, 25 clients, respectively.
3. Use the 1st set of the workload characteristics and double the think and the key time. Obtain the response time and the transaction throughput for 5, 10, 15, 20, 25 clients.
4. Use the 2nd set of the workload characteristics and double the think and key times. Obtain the response time and the transaction throughput for 5, 10, 15, 20, 25 clients.

Experiments for both implementations used the same hardware configuration of low-end platforms consisting of three 133 MHz Intel PCs with 32 MB RAM, and one 233 MHz Intel PC with 128 MB EDO SIMMS and EIDE hard disks. The 233 MHz PCs was running both the web server and also the local DB server. Note that generally it is not recommended to house both the web server and DB server on one system. The other PCs were running supplier DB servers and the bank's DB server. Each client was running on a separate 133 MHz PC. The network configuration was identical in both cases as well, consisting of one local area network. The experiments were performed in a controlled environment while no other applications but the benchmark experimentation was present. Furthermore, for both implementations the same benchmark generator was used, as was the same RDBMSs. The difference between the two implementations was the middle-tier layer. One implementation was based on COM with the components implemented using Visual Basic, while the other one was based on CORBA with the business logic implemented using C++ classes. [Wang 1998] and [Cao 1999] present in detail the hardware and software platforms in terms of detailed-level of specifications (e.g. chip set description and version numbers).

### 5.3.1 Results for COM Implementation

Figures 5.1 and 5.2 illustrate the respective throughput and average response times for 5 to 25 users. The throughput is almost twice as high in cases 1 and 2 as opposed to 3 and 4 that are for “slower” users! The throughput is thus highly dependent on the user behavior. The distinction between case 1 and 2 is that the former is more write oriented, containing more of the buy transactions that stress the system and lead to lower throughput than case 2. Similar result is obtained for cases 3 and 4 of slower users. Also note that the throughput appears to peak for 20 users and any additional users actually cause decline in throughput! The reason is the arrangement of housing both the web and DB server on one system. The low-end platform was

able to handle 20 clients successfully, but any further increase in the number of clients causes over-load particularly due to DB servers' demands on CPU and disk resources and thus cause over-all performance degradation.

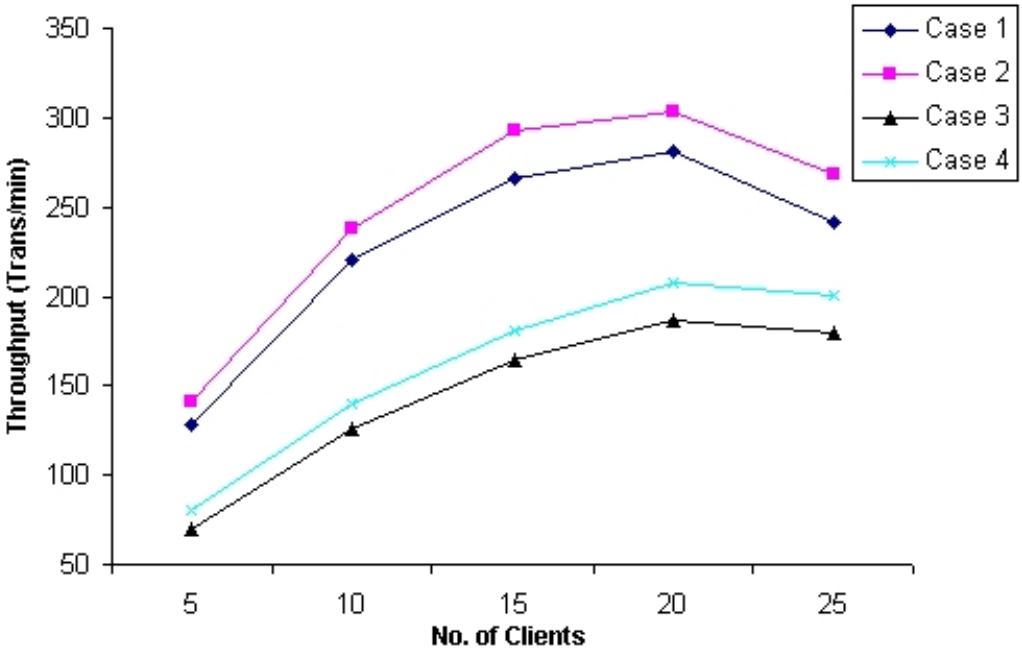


**Figure 5.1** Throughput and Number of Clients for COM Implementation

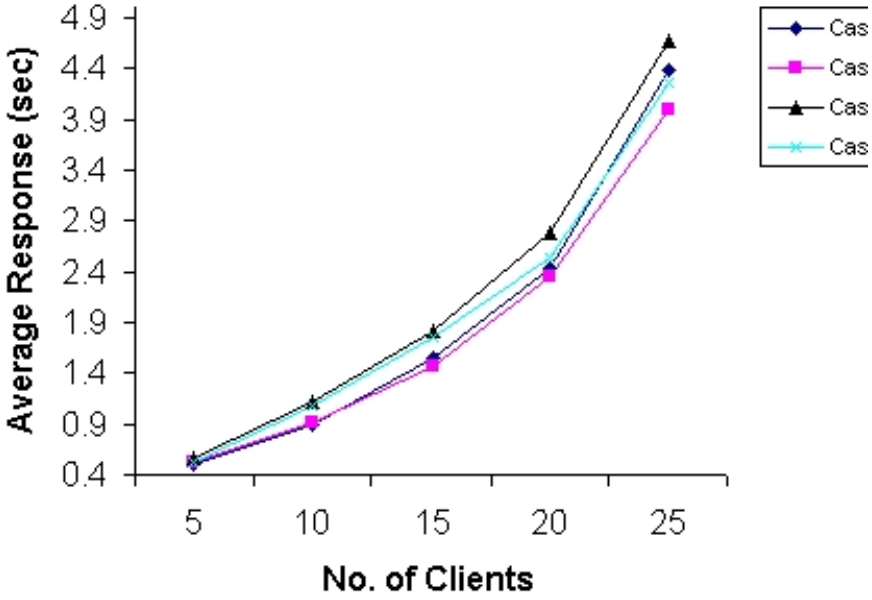


**Figure 5.2** Average Response Time for COM Implementation

Figure 5.2 shows the average response time. The differences between cases 1 to 4 do not appear as profound as was the case for throughput. When moving from 20 to 25 users there is a sharper rise in the response time than is in range from 5 to 20. The reason again is that at 25 clients the system is overloaded.

### 5.3.2 Results for CORBA Implementation

Figures 5.3 and 5.4 illustrate the respective throughput and average response times for 5 to 25 users. The throughput rises almost in a linear fashion. Cases 1 and 2 are for "faster" users, that is for cases with the quicker think and key times than in cases 3 and 4 of slower users. For the two cases of faster users, higher throughput is obtained for case 1 containing less writes

to DBs. Cases 3 and 4 for slower users are similar. The distinction between the slow and fast users is not as pronounced as for the COM implementation.

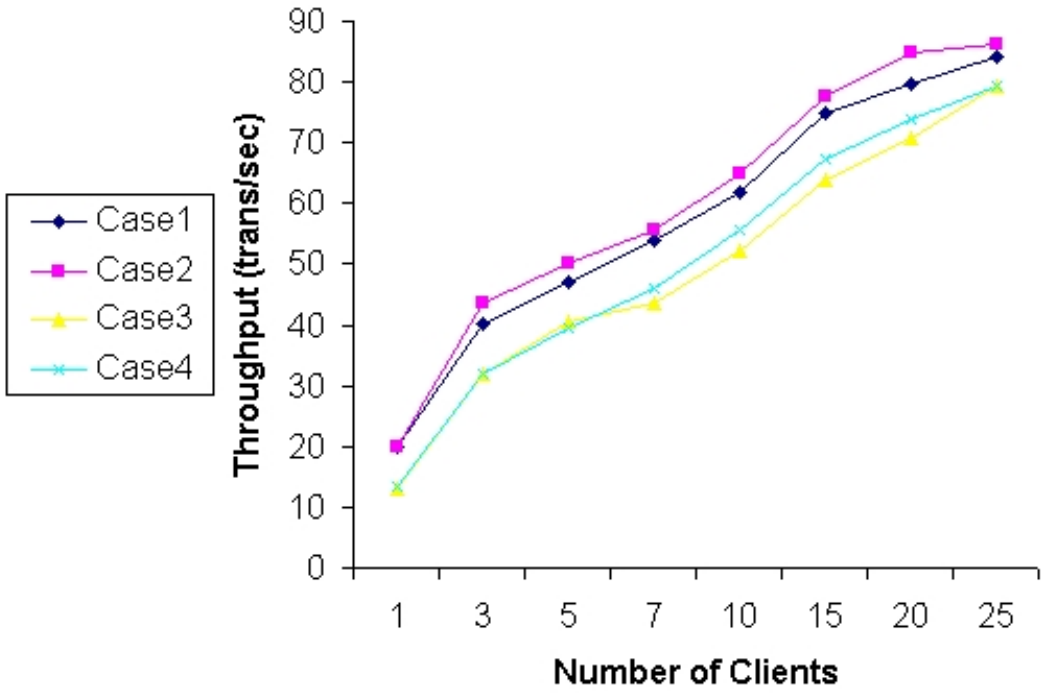


**Figure 5.3** Throughput for CORBA Implementation

The response time, shown in Figure 5.4, increases in what appears to be linear fashion with the number of clients. There do not appear to be differences due to think and key times nor due to variation in the read/write activities of transactions.

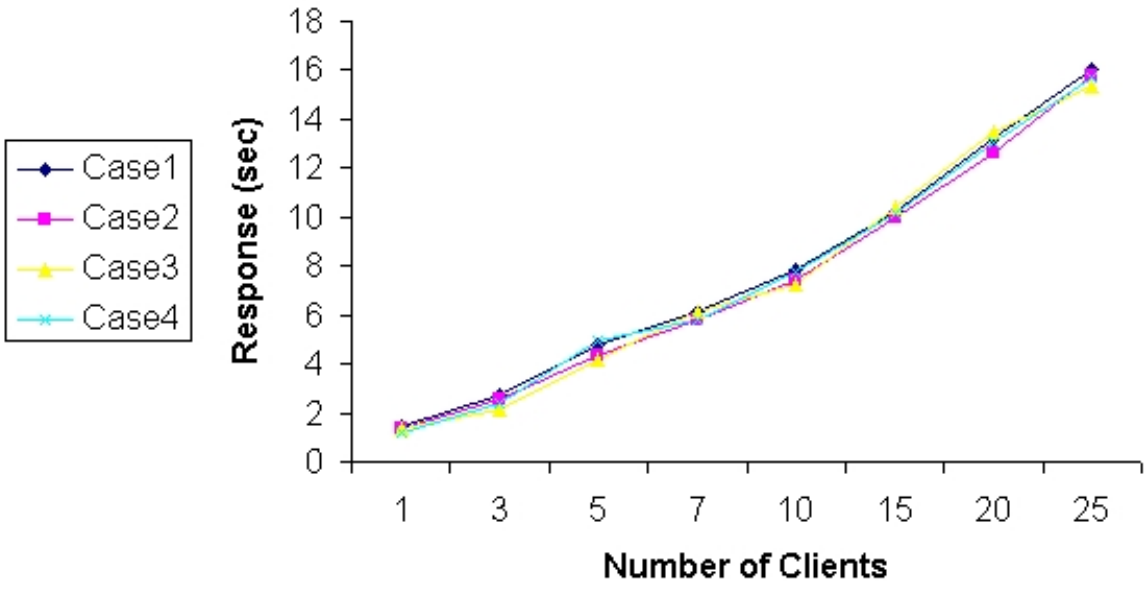


**Figure 5.4** Response time for CORBA Implementation

Closer examination of the delays revealed that the OTS causes substantial overhead delays that have detrimental affects on the system throughput. Because of the relatively low number of transactions stressing the local DB server and the web server, both housed on one PC, over-loading and consequent decrease in throughput is not exhibited even at 25 clients.

## 5.4 Comparison of Implementations

Figures 5.5 and 5.6 show the respective throughput and response time for the COM and CORBA implementations. In both cases the first four bars are for the COM cases while the last four bars are for the CORBA ones. It is apparent that the overall performance of the COM implementation is better than that of CORBA both in terms of throughput and response time. We offer the following explanations:

- Windows NT Operating system was used. Clearly, interactions between Microsoft products and the NT are far more efficient than those between the NT and third-party products.
- COM components are dlls, which means that they are linked into the address spaces of existing processes while CORBA components are not.
- There is less communication under COM. CORBA uses naming and dispensing services resulting in more messaging between clients and servers.
- Efficiency of the compiled code can have great effect. For the COM implementation, Microsoft's Visual Basic was used. For the CORBA implementation we could not use Microsoft's Visual C++ because of conflicts between Visual C++ and Orbix.

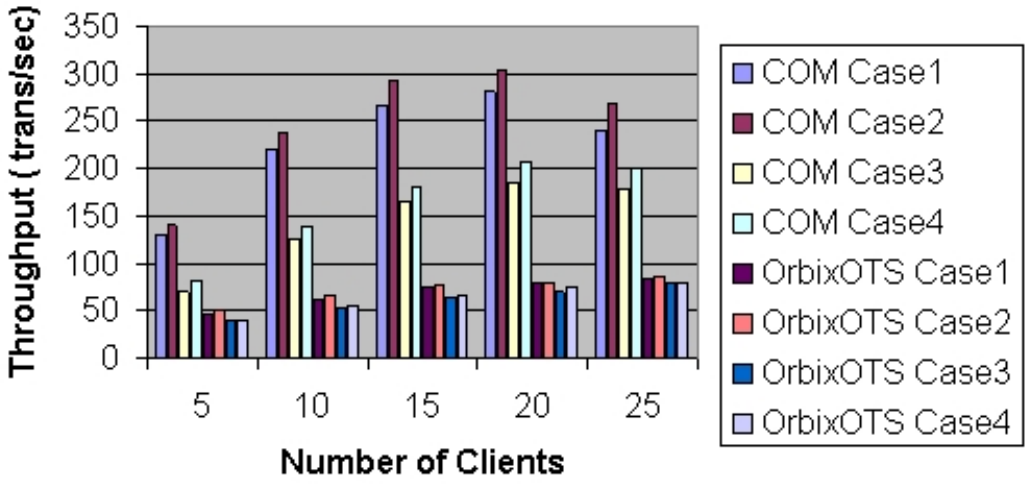


**Figure 5.5** Throughput for COM and CORBA Implementations

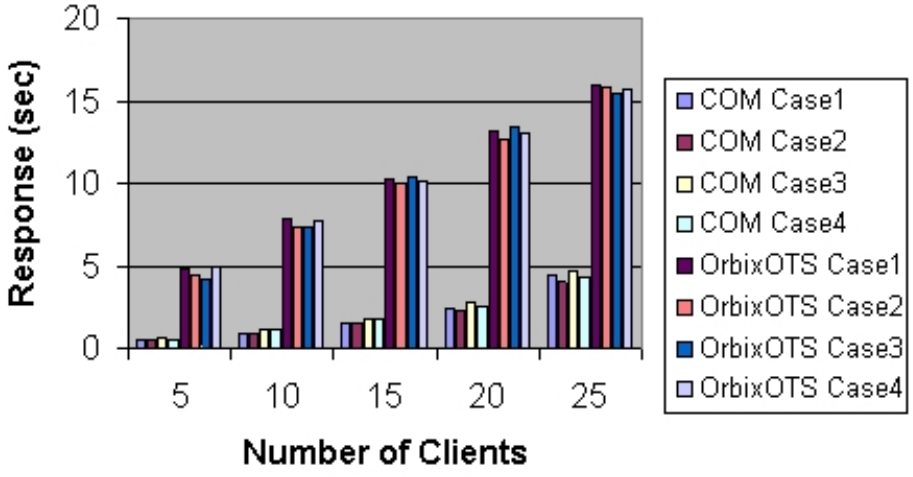


**Figure 5.6** Response Time between COM and OrbixOTS

In spite of danger of repetitive statements, we note again that the same software was used for the User Services, except for the ASP scripts, and Data Services layers of the three-tier architecture. Thus the difference in performance was primarily due to the middle layer for reasons stated above.

Our placement of both web and local DB servers on one low-end system was intentional. We expected to observe saturation of CPU and disk resources on that system as the number of clients increased and thus observe actual decrease in throughput. This indeed occurred for the COM implementation. It did not occur for the CORBA implementation at least not for the range in the number of clients we used. This was due to high messaging overhead in the middle tier.

## 5.5 Experimental Results: Summary and Conclusions

This section examined throughputs and delays for two implementations of the WebEC benchmark. Obviously, we do not claim that the quantitative results are representative in terms of the magnitude of numbers - the experiments were performed in a university lab setting with low-end equipment. The experiments, however, clearly demonstrate the usefulness of benchmarking on two fronts: choice of technology and scalability. When the delays and throughput were compared for the two implementations, COM was clearly superior in terms of efficiencies. This was expected for the particular platforms and development environment. For the COM implementation, the OS, COM/DCOM, and development environment (VC++) are from one vendor and hence are expected to result more efficient. A realistic environment for WebEC would include various platforms on which clients and servers run. In such a scenario, the interoperability gained by using CORBA would be one of the highest priorities.

In terms of scalability, for the limited number of clients tested (maximum of 25), the CORBA implementation did not appear hit bottleneck when scaling up to 25 clients. One might speculate that the throughput appears to be tapering of at 25 clients in Figure 5.3, but further experiments involving a higher number of clients are required. For the COM implementation, on the other hand, the throughput clearly peaks at 20 clients and actually decreases when the number of clients increases to 25. The reasons for this were already discussed above.

On the final note, we used traditional performance metrics, throughput and response time. We are currently addressing subtleties for the throughput definition that involve the perception of when a transaction is completed. For example, is a browse transaction completed when the first character of the results appears on the client or when all results are displayed? Further traditional metrics, the utilization of CPUs and stress on the secondary storage (disks) are being augmented. Even more importantly, we are also addressing issues of quality of service such as video quality, sound quality, availability, and reliability in relation to how they are measurable and defined as benchmark metrics.

# 6. Related Work

An electronic commerce system may be assembled from a variety of web servers, database servers, transaction servers, database connectivity mechanisms, security and payment protocols, and business logic components. Benchmarks for individual components are currently available. Webperf [1997] from Standard Performance Evaluation Corporation (SPEC) is a benchmark for web servers. It calculates throughput in HTTP operations per second and client response times per HTTP operation. WebSTONE [Trent 1995] measures the performance of an HTTP server, which facilitates the evaluation of different implementations of HTTP specification. In WebSTONE, several workloads are used: modem uses, down load times, media rich larger content, etc.

The Transaction Processing Performance Council (TPC) provides a TPC-C benchmark specification that is widely used by industry to compare the performance of database servers on different hardware configurations. The business logic for the TPC-C [TPC 1997] benchmark is based on an order-entry application that may be distributed among many sites. TPC-C measures the performance of RDBMSs in terms of business transaction throughput and cost per transaction. TPC-C addresses many of the shortcomings of TPC-A and TPC-B, the older and now obsolete Industry wide relational database benchmarks. The TPC-D benchmark exercises decision support systems on very large size databases.

The Wisconsin Benchmark [Webperf 1997] systematically measures and compares the performance of relational database systems with database machines. It measures the query optimization performance of database systems with 32 query types to exercise the components of the proposed systems.

Benchmark Factory [1997] provides prepackaged scalable benchmarking and load-testing tools. Implementations of a number of pre-developed industry standard benchmarks including TPC-B, TPC-C, TPC-D and Wisconsin can be bought from Benchmark Factory. The ODBC driver benchmark, BENCH [Geiger 1995], is a software application that is loosely based on the TPC-B benchmark, and which provides a meaningful comparison tool to judge the performances of different ODBC drivers for OLTP application using the same DBMS.

[Krishnamurthy 1997] focuses on developing tools to characterize the performance of e-commerce systems and to carry out scalability analysis and capacity planning exercises. The tools build an information model for the system from which synthetic workloads and predictive performance models can be generated.

TPC [TPC 1998] announced its intent to create a new Electronic Commerce Web Benchmark (TPC-W). The TPC-W benchmark will represent a typical workload of a business-oriented e-commerce solution where there is simultaneous execution of multiple transaction types that span a breadth of complexity. The benchmark is yet to be released.

In [Jutla 1999b], TPC-C is used as the basis of a web transaction processing benchmark application. The benchmark extends the TPC-C for relational database management system to cater for the Web environment. Specifically, the notion of transaction was to include the web browser. [Jutla 1999a] presents business models for e-commerce and details the factors that business must consider in the decision-making process for investment in e-commerce. [Jutla 1998] discusses distributed object technologies for e-commerce applications and overviews benchmarks and total costs and benefits of ownership models for IT investment.

# 7. Summary and Conclusions

This paper described WebEC, a benchmark for the e-broker business model of e-commerce. The benchmark's transactions, databases, navigational layout, security, and mix of transactions were described. Implementations using Microsoft's COM technology and also CORBA technology were also described. Under the specific experimental environment the COM implementation was superior in performance. The environment may be suitable to mimic the e-commerce behavior of a small enterprise. It was also observed that COM performance peaked at 20 clients and actually declined with further increase in the number of clients. COM implementation performance was also sensitive to the user behavior. The simple experiments demonstrated the value of benchmarking when examining the system under consideration for scalability and also for the choice of technology.

An e-commerce platform can be assembled from various combinations of web servers, transaction servers, commerce servers, web browsers, security protocols, ODBC implementations, database connectivity solutions, operating and DBMS systems. E-commerce benchmarks are of value to IT professionals and managers in the decision making process for final IT investment. The benchmarks aid IT managers in determining the number of users that a system can support, system response times under various loads, and estimations of scalability. These are important decisions for businesses about to invest in Intranets, Electronic Commerce and other web applications.

The various available database connectivity solutions restrict the portability of a Web benchmark implementation. An enterprise should match the benchmark’s connectivity solution with the same technology used in their real Web applications. An enterprise must match the middleware solution to the same technology that is to be used in the real Web applications. Also, the distributed object models (e.g. CORBA or COM), selected by the business for use in its distributed web applications such as e-commerce, must match.

The two implementations of the WebEC comprise first steps towards a development of a repository of benchmarks for e-commerce. The repository may be viewed as an initial tool towards selecting appropriate technology and platforms and examining potential performance and scalability of chosen solutions.

# References


**Benchmark Factory 1997**; Benchmark Factory, http://www.benchmarkfactory.com (1997)

**Cao 1999**; Cao Lihong., "CORBA Implementation Of the WebEC", Masters of Computer Science project, Faculty of Computer Science, Daltech, Dalhousie University, Halifax, Nova Scotia, Canada, December 1999.

**Francis 1998**; B. Francis, A. Fedorov, R. Harrison, A. Homer, S. Murphy, D. Sussman, R. Smith, S. Wood, **Professional Active Server Pages 2.0**, Wrox Press Ltd., Birmingham, UK (1998)

**Geiger 1995**; K. Geiger, “Inside ODBC,” Microsoft Press, Seattle, Washington, USA, (1995)

**Jennings 1997**; R. Jennings, Database Workshop – Microsoft Transaction Server 2.0, SAMS Publishing, Indianapolis, Indiana (1997)

**Jutla 1998**; D. Jutla, P. Bodorik, Investing in E-Commerce, 2nd Annual International Conference on Emerging Issues in Business and Technology, Myrtle Beach, South Carolina, November 12-14, 1998, pp. 14-23 (1998).

**Jutla 1999a**; D. Jutla, P. Bodorik, C. Hajnal, C. Davis, Making Business Sense of Electronic Commerce, IEEE Computer, March 1999, pp. 65-76.

**Jutla 1999b**; Jutla, D.N., Bodorik, P., Ma, S., Wang, Y., WebTP: A Benchmark for Web-based Order Management Systems, 32nd Annual Hawaii International Conference on System Sciences, Web Information Systems track, (HICSS'99), Jan 5-9, 1999, (10 pages).

**Krishnamurthy 1998**; D. Krishnamurthy, J. Rokia., Workload Characterization Tools for E-Commerce Servers, Electronic Commerce, International IFIP/GI Working Conference on Trends in Distributed Systems of Electronic Commerce, pp.5-15, dpunkt.verlag, Hamburg (1998).

**Riggins 1998**; Riggins, Frederick and Rhee, Hyeun-Suk, Toward a Unified View of Electronic Commerce, Communications of the ACM, October 1998, Vol. 41, No. 10.
**Rogerson 1997**; D. Rogerson, Inside COM, Microsoft Programming Series, Microsoft Press, Washington (1997)
**TPC 1997**; TPC Benchmark C, Standard Specification, Revision 3.3.2, Transaction Processing Performance Council (TPC), http://www.tpc.org (1997)
**TPC 1998**; TPC Electronic Commerce Web Benchmark (TPC-W Benchmark) Announcement Overview, Transaction Processing Performance Council (TPC), http://www.tpc.org (1998)
**Trent 1995**; Trent, G. and Sake, M., "WebSTONE: The First Generation in HTTP Server Benchmarking," White paper, MTS Silicon Graphics, (1995)
**Wang 1998**; Wang Y., "WebEC: An Electronic Commerce Benchmark", Masters thesis, Faculty of Computer Science, Daltech, Dalhousie University, Halifax, Nova Scotia, Canada, December 1998.
**Webperf 1997**; Webperf, http://playground.sun.com/pub/prasadw/webperf, (1997)
**Wisconsin Benchmark 1998**; Wisconsin Benchmark, http://benchmarkresources.com/ handbook/4-1.html