Measurement-based Performance Analysis of E-commerce Applications with Web Services Components

Venu Datla
Katerina Goseva–Popstojanova

Presented at the IEEE Conference on e-Business Engineering (ICEBE 2005), Beijing, China, October 2005

This work was funded by NSF CAREER grant number CNS-0447715 and NASA OSMA SARP grant managed through NASA IV&V Facility

---

E-commerce applications & Web services

- Modern e-commerce applications are large-scale, distributed and depend on various inter-enterprise and intra-enterprise services for execution
  - Integration with the application becomes complex task since these services are often developed on different platforms, using different programming languages and technologies

- The promise of Web services
  - Interoperability
  - Dynamic and flexible application integration
  - Loose coupling
  - Service oriented architecture

---

What are Web services?

- **W3C Definition**: A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format.
  - **Web Service Description Language (WSDL)** is an XML grammar for specifying the properties of a Web service (messages exchanged by the service, operations supported by the service, protocol bindings and endpoints of the service, etc)
  - **Universal Discovery, Description, and Integration (UDDI)** provides a standard way for businesses to publish and discover Web services
  - **Simple Object Access Protocol (SOAP)** is a standard for sending messages and making Remote Procedure Calls over Internet. It uses XML for data encoding and HTTP as the transport protocol
  - Other transport protocols (FTP, SMTP, or even TCP/IP sockets) may be used
Web services communication stack

Overheads:
• Marshalling and unmarshalling between objects and XML elements
• Time required to encode and parse SOAP messages
• Time consumed in encrypting and decrypting XML messages for Web services using security

Motivation

• Performance is an important quality attribute of Web services
  – Only a few papers have focused on performance evaluation of Web services in the past
    • throughput and overall system response time

Contributions

• Measurement-based performance study of a three-tier e-commerce application which includes Web services components
  – at software architectural level we instrumented the application to log components’ response times in Application server logs
  – at hardware resource level we used Windows performance monitor to capture the hardware resource level performance
• Implemented a session oriented workload generator based on TPC-W specification
  – Measured application performance under different workload profiles and different workload intensities
Benefits

• Our empirical results contribute toward quantifying the overhead introduced by Web services
• Help identifying software components and hardware resources which are bottlenecks in the system
• For example, we show that Web services components have significantly higher response time under the Ordering profile
  – This information is important for system designers since customers in Ordering profile tend to have more ordering activity and generate revenue

Prototype description

• We developed a prototype of an e-commerce application
  – A travel agency that offers flight booking services to its customers and uses secure transfer for confidential data
• The prototype has a three-tier architecture
  – user interface layer (JSP Web pages)
  – business logic layer (EJB components, Web services components)
  – data layer (Oracle database)
• The application is implemented using J2EE technologies which facilitate development of scalable, robust, multi-tiered enterprise systems

Software architecture

• The user interface layer consists of a set of Web pages
  – Home, Search, Search results, Shopping cart, Customer login, Check credit, Process orders
• The business logic layer contains components that implement the core functionality of the travel agency application
  – Flight-WS: Web service that searches for flight schedule information
  – Currency-WS: Web service to compute exchange rate between currencies
  – Credit-WS: Web service to check customer credit information
  – Customer-EJB: EJB component which contains customer information
  – Login-EJB: EJB component to authenticate customers
  – Order-EJB: EJB component to store order information
• The data layer consists of a backend relational database management system which stores the persistent information in the form of database tables
Development & Deployment

• We have developed all Web services components and hosted them locally
  – The first version of our prototype integrated publicly available Web service for the Flight-WS
    • this service had low availability
    • when it was available, the server responded with server error whenever more than five simultaneous search request were generated
  • This does not seem to be an isolated incident
    – A study conducted in 2001 showed that 48% of the production UDDI registry had links that were unusable
    – A more recent study reported similar findings - over the period of six months (Aug 2003 to Jan 2004) 67% of Web services registered in the UDDI registry were invalid (their WSDL files were either inaccessible or not registered)

Development & Deployment

• By hosting all software components locally we avoid accounting for network latency which is beyond our control
  – This does not limit the scope of our research since our goal is to study the contribution of software components to the response time of e-commerce interactions at server side rather than to study end-to-end response time as perceived by the user
  – Even more, hosting all components locally supports experiments with higher workload which may not be possible with publicly hosted Web services components

Deployment diagram
Implementation details

- We use J2EE, a widely used standard which facilitates development of scalable, robust, multi-tiered enterprise systems
- User interface layer
  - Java Server Pages (JSP) which is a J2EE technology for creating dynamic Web content
  - Tomcat v5.0 as a Web server
- Business logic layer
  - Web services and Enterprise Java Beans (EJB)
  - J2EE 1.4 Application Server Platform Edition 8.0
- Data layer consists
  - Oracle 9i Release 2

Deployment details

- Workload generator runs on a laptop: 1.2 GHz Pentium M processor with 512 MB RAM
- Web server and EJB components run on the Application server 1: 3 GHz Pentium 4 processor and 1 GB RAM
- Web services components run on the Application server 2: 3GHz Pentium 4 processor and 1GB RAM
- Database server runs on a different machine with the same configuration and 120 GB disk drive
- All machines run Windows 2000 operating system and are connected through Ethernet LAN with 100 Mbps speed

Why experiments with synthetic workload?

- The most realistic way to study performance is under the real workload. However,
  - the real Web workload is complex, cannot be controlled and is not repeatable
- Alternative – synthetic workload based on performance benchmarks
  - Measurable and repeatable
  - Enables comparative studies of products
  - Vendors, developers, and users run benchmarks to accurately test new systems, pinpoint performance problems, or assess the impact of modifications to a system
Workload generation: TPC-W

• TPC-W is a transactional Web benchmark which specifies an online bookstore e-commerce application
  – simulates activities of a business oriented transactional Web server
  – follows a closed loop model: a client sends new request only after receiving response to the previous request
  – session oriented workload
  – three workload profiles with different percentages of browse and buy operations
    - Browsing
    - Shopping
    - Ordering

Workload characterization in TPC-W

• We have implemented the workload generator based on TPC-W specification for our prototype
• The workload is modeled with a Discrete Time Markov Chain (DTMC) which characterizes users’ request pattern within a session
  – In TPC-W it is called Customer Behavior Model Graph
• This represents the user view on the system

TPC-W profiles

• We conducted measurements and analyzed results for two workload profiles: Browsing and Ordering

<table>
<thead>
<tr>
<th></th>
<th>Browsing profile (79-21)</th>
<th>Ordering profile (59-59)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Home</td>
<td>21</td>
<td>17</td>
</tr>
<tr>
<td>Search</td>
<td>30</td>
<td>17.5</td>
</tr>
<tr>
<td>Search Results</td>
<td>28</td>
<td>15.5</td>
</tr>
<tr>
<td>Shopping Cart</td>
<td>12</td>
<td>14</td>
</tr>
<tr>
<td>Customer login</td>
<td>3.2</td>
<td>13</td>
</tr>
<tr>
<td>Check credit</td>
<td>2.9</td>
<td>11.5</td>
</tr>
<tr>
<td>Process order</td>
<td>2.9</td>
<td>11.5</td>
</tr>
</tbody>
</table>
Other TPC-W specifications

- Three phases: rump-up, steady-state, and ramp-down
  - The data is collected during the steady-state interval with duration of 30 minutes
- The size of the database must be scaled accordingly to the number of clients
  - We ran experiments with 50, 100, 150, and 200 clients
  - Following the TPC-W specification we populated the database with a customer table with 576,000 rows
- The average think time is 7 seconds and the average user session duration is 11 minutes
- For each Web interaction, the TPC-W measures at the client side (i.e., Emulated Browser) the Web Interaction Response time (WIRT)

Measurement methodology

- We instrumented the application to capture the performance data for components running on different machines
  - Service layer – software architectural view
  - System layer – deployment view
- Each request involves processing by one or more components
  - Search request: Search page
  - Shopping Cart request: Shopping Cart page
  - Customer Login request: Customer Login page
  - Home request: Home page and Customer-EJB
  - Search Results request: Search Results page and Flights-WS
  - Check Credit request: Check Credit page, Login-EJB, and Currency-WS
  - Process Order request: Process Order page, Credit-WS, and Order-EJB

Component execution times are logged in the Application server logs using java.util.logging API

Application server log record format

```
{yyyy-mm-ddThh:mm:ssZ | [logLevel] | className] [methodName | [KeyId] Value Pair: [MessageId: Message]
```

- We used scripts written in AWK scripting language to parse the Application server logs and extract response times of components participating in each request
90th percentile response time measurements

Analysis of the results

- Response time increases with workload (number of customers) for both profiles.
- Response times for 50, 100, and 150 clients are approximately the same for the Browsing and Ordering profiles.
- In case of 200 clients response times for the Ordering profile are approximately 20% higher than for the Browsing profile.
- Critical requests (user layer): Search results, Process order, and Check Credit.
- Critical components (software architectural layer): Flight-WS, Currency-WS, and Credit-WS.
  - They have higher response time than other components (60-80% of the overall response times of Search results, Process order, and Check Credit is spent in executing Web services) and are more sensitive to the workload.
  - This is due to the time used for parsing the XML requests and responses and the higher database activity.

Measurement methodology

- For hardware resource monitoring we used Windows 2000 performance monitoring tool.
- On each machine we recorded:
  - The percentage of non-idle processor time spent in user mode (%User Time).
  - The rate of read and write operations on the disk (Disk Transfers/sec).
Results: Hardware resource measurements

CPU utilization increases significantly for 200 clients

One of the reasons for increased response time of Web services components under higher workload is the overhead due to parsing and encoding the XML messages which leads to increased CPU utilization

EJB and Web services components perform operations on the backend database

Disk transfers per second for ordering profile are higher than for browsing profile. The difference increases with the workload intensity, which explains the increase in response times of EJB and Web services components.

New version of the application server

- We deployed the same implementation of the prototype on a newer version of the application server, J2EE 1.4 Application Server - Platform Edition 8.1
  - This server was the latest available production version in December 2005
- The Web server, database server, operating systems, and the hardware configuration remain the same
- The results show that the new version of the application server improves the response times of Web interactions significantly
  - Since Web service components experience the most dramatic improvements, we present more details for the Search Results, Credit Check, and Process Order interactions which include Web service components to fulfill the user requests

90th percentile response time measurements under the new application server

Response time of the Web services components is significantly more sensitive to the workload than the response time of other components under the new application server as well, that is, the Web services components still scale worse than the other components

Also, the response time of the Web services Components is higher in the Ordering profile than in the Browsing profile, especially for higher workload (i.e., 150 and 200 customers).
Response times for Flight-WS normalized with respect to response time for 50 customers

Platform Edition 8.1 application server improves the response times of Web services components, as well as their scalability.

Response time improvement for Flight-WS component

J2EE 1.4 Application Server Platform Edition 8.1 improves the response time of Flight-WS component from 11% to 44%.

The improvement increases with the workload intensity which leads to better scalability of the Web services components.

The amount of improvement is not very sensitive to the workload profile.

The results for Currency-WS and Credit-WS components are consistent with these observations: response times of Currency-WS and Credit-WS components improve 1% - 43% and 0% - 35%, respectively.

Conclusion

• In contrast to the related work which evaluated the overall application response time, our study includes measurements and analysis of the server-side performance at different levels:
  - Web request level allows us to study the user point of view.
  - Software architecture level allows us to study the distribution of the Web request response time among different components used to process the request.
  - Hardware resource level provides additional insights and helps explaining the observed phenomena.

• Enables system designers and application developers to improve the performance in a cost effective manner.
Conclusion

- Web services tend to become bottlenecks of the system, particularly under heavier load
  - The phenomenon is attributed to the overhead introduced by additional processing of XML messages and basically, is the price paid for interoperability
  - Possible solutions
    - develop more efficient XML parsers
    - Incorporate better mechanisms for encoding and decoding of SOAP messages
- The response time of the Ordering profile under higher workload becomes significantly worse than the response time of the Browsing profile
  - Main reasons: higher database activity and contention for database resources which affect the performance of EJB components and even more the performance of Web services components

Conclusion

- Software architectural level and hardware resource level measurements provide essential information, such as identification of critical software components and hardware resources, that can be used by system designers and application developers to improve the product performance
- Our latest results emphasize the importance of the server-side infrastructure selection, in addition to the software architecture selection, because both can profoundly affect the resulting performance and scalability of the application
  - For our prototype, using the newer version of the J2EE 1.4 application server led to improvement of Web services response times up to 40% under higher workload
  - It follows that a simple upgrade of the server-side infrastructure, without additional development and implementation efforts, is a cost effective way to provide significantly improved performance and scalability levels