Comparing Industry Benchmarks for J2EE Application Server: IBM's Trade2 vs Sun's ECperf

Yan Zhang
School of IT,
University of Sydney,
NSW 2006, Australia
yanzhang@cs.usyd.edu.au

Anna Liu
Software Architectures and Component Technologies Group, CSIRO Mathematical and Information Sciences,
Locked Bag 17, North Ryde, NSW 2113, Australia
Anna.Liu@csiro.au

Wei Qu
University of Wollongong,
NSW 2522, Australia
wq03@hotmail.com

Abstract

As the Internet and enterprise wide distributed systems become more prevalent in business IT systems, numerous advanced COTS (commercial off-the-shelf) middleware technologies have appeared on the market. One such leading middleware technology type is Sun's Java 2 Enterprise Edition (J2EE) technology. At present, there is an abundance of J2EE application server implementations in the marketplace with almost no discerning differences. The different product vendors all claim to have implemented the highest performance and most scalable product. The challenge for the potential J2EE application server user is then in choosing the right product for their purpose.

In order to select a J2EE application server from the confusing J2EE product market, lots of organizations defined their own test suite and carried out their evaluation of J2EE product using their benchmark applications. Unfortunately evaluating application servers is a costly exercise, and proprietary benchmarks and corresponding results are not often reusable for different organizations. To alleviate this problem, there are emerging industry J2EE benchmarks that aim to produce reusable benchmark results.

In this paper, we provide insights into performance benchmarking of J2EE technologies via critically evaluating two such industry J2EE benchmark standards: IBM's Trade2 benchmark application and Sun's ECperf. We firstly aim to share the experience gathered during the empirical experiment in using the benchmarks. These insights obtained and shared will provide assistance in future performance benchmark design, as well as informing the users how to be more discerning when reading published benchmark results..

Keywords: benchmark, middleware, empirical results, COTS, component-based system

1 Introduction

To date, more and more business information systems are being web-enabled due to the popularity of the Internet and middleware technologies. The Internet technology provides a communication medium for businesses and their potential customers. The middleware technology hides the low level networking details away from the programmer and provides solutions that are robust, scalable and high performance for e-businesses. Sun's Java 2, Enterprise Edition (J2EE) component technology is one of the most technically advanced COTS (commercial off-the-shelf) middleware technologies on the market. The J2EE specification describes a server-side component based way of developing distributed Java systems. The J2EE framework thus simplifies the development of complex distributed systems by providing ready to use enterprise services such as transactions and security. Thus, J2EE is becoming widely adopted in business systems and its integration with Internet-enabled components makes it attractive as a back-end platform for many e-business systems.

However, a J2EE-based application component's performance is dependent upon the actual J2EE application server product that it runs on (Chen, S., Gorton I., Liu A., and Liu Y. 2002). The same application component may perform very differently indeed on two different J2EE application server implementations, depending on the quality and features of the application server product. Currently, there are over 40 known implementations by different product vendors (Gorton I. and Liu A. 2002). Each vendor claims superior features, performance and functionality but varying in cost. This becomes a significant challenge for the selection of the right product for the business. In response to these challenges faced by the businesses, lots of independent organizations defined their own test suite and carried out evaluation of J2EE product using their benchmark applications. Unfortunately evaluating application servers is a costly exercise, and proprietary benchmarks and corresponding results are not often reusable for different organizations. To alleviate this problem, there are emerging industry J2EE benchmarks that aim to produce reusable benchmark results.

In this paper, we provide insights into performance benchmarking of J2EE technologies via critically evaluating two industry standard benchmarks: IBM's Trade2 benchmark application and Sun's ECperf. Section 2 gives an overview of IBM's Trade2 and Sun's ECperf followed by section 3 which reports some sample results on evaluating IBM's Websphere using these industry standard benchmarks; we then compare the strengths and

weaknesses of Trade2 and ECperf in section 4; section 5 concludes the paper.

The purpose of this paper is to share the experience gathered during the empirical experiment, and to highlight that it is important to understand the motivations and test setups of benchmarks while reading and interpreting the corresponding published benchmark results. These insights obtained will provide assistance in future performance benchmark design, as well as informing the public of the importance of reading test results with discerning eyes.

2 Overview of IBM's Performance Benchmark Sample and Sun's ECperf

2.1 IBM's Performance Benchmark Sample

IBM's Performance Benchmark Sample was originally developed by the IBM WebSphere Performance team to test the performance of the WebSphere Application Server. It provides a suite of workloads to characterize the performance of the WebSphere Application Server. The workloads include Web Primitives and Trade Application (Trade2).

Web Primitives are used to individually test various components of the WebSphere Application Server such as the Servlet engine, JSP (Java Server Pages) support, EJB (Enterprise Java Beans) entity and Session Beans, HTTP session support and so on.

Trade2, which was derived from experience with many customer environments, models an online brokerage. It is a collection of Java classes, Java Servlets, Java Server Pages and Enterprise Java Beans built to open J2EE APIs to provide web based services such as login/logout, stock quotes, buy, sell, account details etc through a standards based HTTP protocol. Figure 1 shows the Trade application model-view-controller architecture (IBM WebSphere Performance Benchmark Sample 2002). In Figure 1, WPBS is abbreviation for Websphere Performance Benchmark Sample and CMP is abbreviation for Container Managed Persistence.

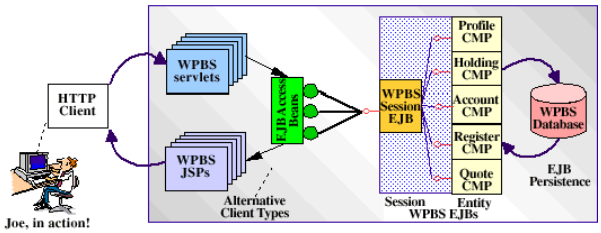

Figure 1: Trade2 Architecture

From a user's perspective, the following actions are allowed via a web browser:

- Register: A person wishing to register with Trade2 can create his/her user profile, ID/password and initial account balance
- Login: A user login to his/her already registered account by inputting ID and password.
- Browse: A user can browse current stock price for a ticker symbol or browse his/her portfolio, etc.
- Update account: A user can modify his/her allocated credit limit.
- Buy: A user can place an order to buy a given number of a specified stock.
- Sell: A user can place a request to sell a specified number of any stock item from holdings
- Logout: A user signs off to terminate his/her activity.

2.2 Sun's ECperf

ECperf (ECperf specification 2001) is an Enterprise Java Beans (EJB) benchmark, which was built by Sun Microsystem with the assistance of the Java community to measure the scalability and performance of J2EE servers and containers. It includes:

- A benchmark application and associated drivers, written in the Java programming language for the J2EE platform.
- A detailed specification for testing and submitting results

The benchmark application models a large global enterprise's activities such as order processing, Just In Time manufacturing, supplier management and corporate functions. It consist of the Enterprise JavaBeans, a set of Java Server Pages for single user interactive testing, schema scripts and load programs to create and load the databases and driver programs to implement the run rules and simulate the client load. All Java source code and make files are included in the ECperf benchmark package. Figure 2 shows the ECperf architecture.

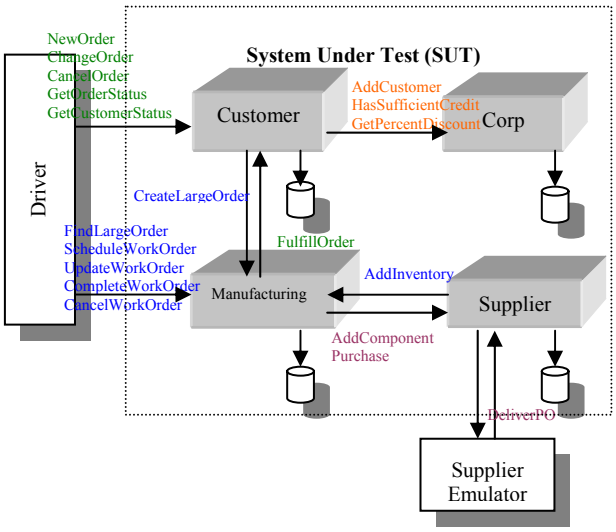

Figure 2 : ECperf Architecture

As shown in Figure 2, ECperf application consists of four application domains: customer, corporate, manufacturing and supplier. There are producer-consumer relationships between domains in the company and to outside suppliers

and customers as well. The workloads in each domain are as follows:

Corp domain workload:

- AddCustomer
- HasSufficientCredit
- GetPercentDiscount

Customer domain workload:

- NewOrder
- ChangeOrder
- GetOrderStatus
- GetustomerStatus
- AddCustomer
- CancelOrder
- ValidateCustomer
- AddToCart
- BuyCart

Manufacturing domain workload:

- ScheduleWorkOrder
- UpdateworkOrder
- CompleteWorkOrder
- CreateLargeOrder
- FindLargeOrders
- CancelWorkOrder
- AddInventory

Supplier domain workload:

- AddComponent
- Purchase
- DeliverPO

Drivers and Supplier Emulation are used to simulate customers placing orders, manufacturing facilities producing products and suppliers providing goods and services. They do not belong to the part of the system under test (SUT). The drivers are in charge of recording all relevant statistics and printing the reports from the run.

3 Test Setup and Sample Results

The following benchmark results are based on IBM WebSphere Application Advanced Single Server V4.0. The test platform comprised of three quad-processor Windows 2000 server PCs, one for client, one for application server and one for database. These machines are connected over 100MB/s switched LAN.

3.1 IBM's Trade2 Test Setup and Sample Results

3.1.1 Test Setup

Here we use Microsoft WAS web stress tool (Microsoft web stress tool 2002) as the web load generator. The web load generator is responsible for firing the TradeScenarioServlet to drive Trade2 (see Figure 3). The web load generator emulates lots of users hitting the web page to perform all kinds of Trade2 operations via the HTTP protocol.

Trade2 provides us with options in workload mixes. A workload mix determines the runtime workload mix of servlets and JSP calls (no database access calls), versus EJB and JDBC calls (database access calls). These mixes include "Standard", "DB-heavy", "DB-light", "Uniform" and "No Register". The "Standard" mix provides a mix of operations similar to typical customer environments. The DB-heavy mix invokes relatively more database intensive operations such as buy and sell. The DB-light setting uses less database intensive operations. The "Uniform" setting drives each operation in equal percentages and finally, the "No Register" mix is the same as the Standard mix except with no new user registration.

Further, Trade2 provides us with run time mode option which determines the mode of database access in the Trade application, that is, the choice between using Enterprise Java Beans or direct database accesses using JDBC. The first mode is "EJB-DB access" which uses EJB technology leveraging IBM VAJ (VisualAge for Java) EJB Access Beans for EJB access. The second mode is "JDBC-DB access" which accesses the database through direct JDBC code and does not use EJBs at all. The third mode is "EJB-ALT-DB access" which uses direct access without leveraging EJB Access Beans.

We refresh the test database by reloading 'TradeDB' before each Trade test run.

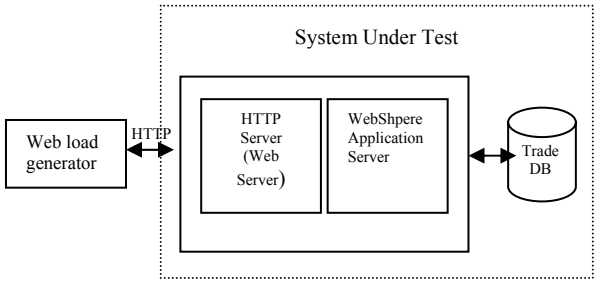

Figure 3: Trade2 Test Setup

3.1.2 Sample Results

Figure 4 shows the result of "EJB-DB access" mode, Figure 5 shows the result of "JDBC-DB access" mode and Figure 6 shows the result of "EJB-ALT-DB access" mode. The requests per second are shown in diagram

(a)'s and the TTFB Avg and TTLB Avg [1] are shown in diagram (b)'s.

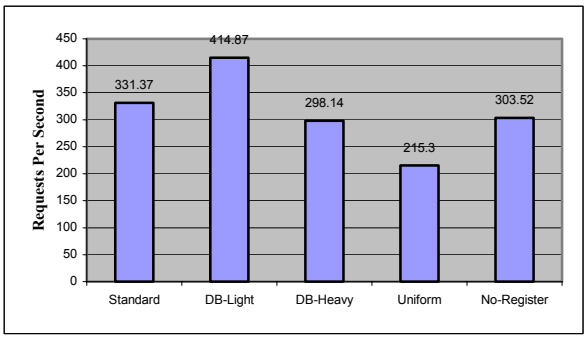

Figure 4 (a)

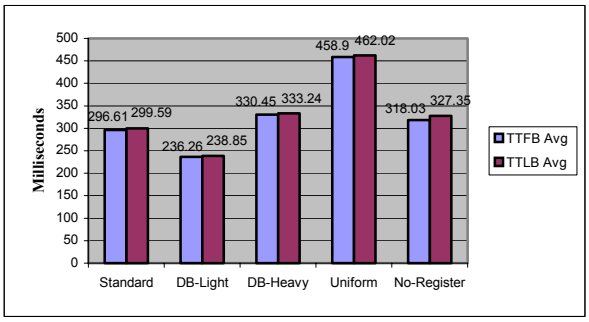

Figure 4 (b)

Figure 4: "EJB-DB access" mode

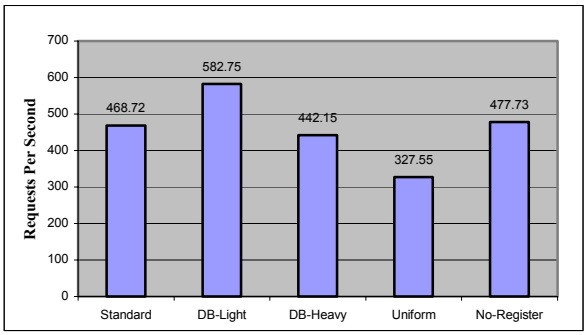

Figure 5 (a)

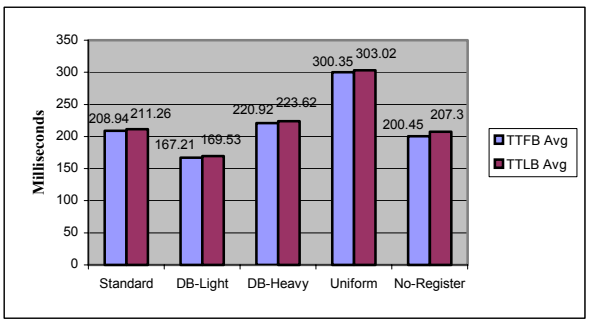

Figure 5 (b)

Figure 5: "JDBC-DB access" mode

[1] TTFB (Time to First Byte) calculates the time from the request for the page until WAS receives the first byte of data, in milliseconds. TTLB (Time to Last Byte) calculates the total time from the request until the last byte of data has been received on the client, in milliseconds. (Microsoft web stress tool 2002)

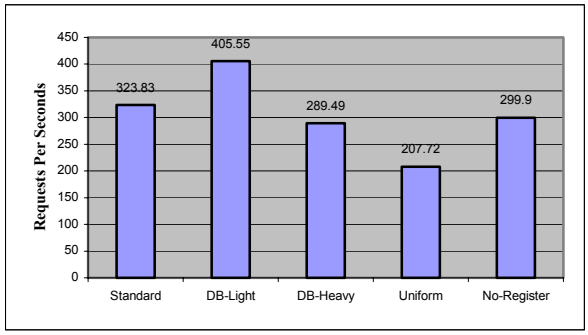

Figure 6 (a)

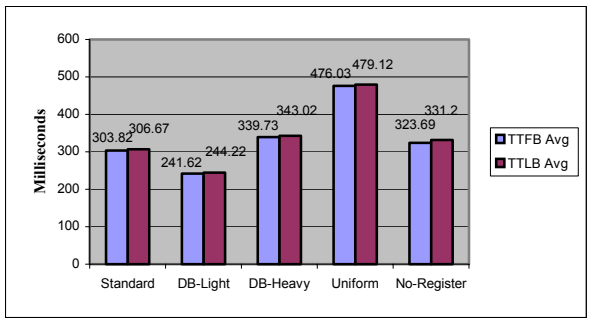

Figure 6 (b)

Figure 6: "EJB-ALT-DB access" mode

From the above results, we can make the following observations:

- The throughput (Requests per second) for the "JDBC-DB access" mode is higher than those of the other two modes. Correspondingly, the TTFB Avg values and TTLB Avg values for the "JDBC-DB access" mode are lower than those of the other two modes. This is most likely due to the extra level of indirections introduced by the entity beans, and additional entity-bean related overheads. There is clearly a performance penalty to pay in the use of entity beans.
- The throughput (Requests per second) for "EJB-DB access" mode is higher than that of "EJB-ALT-DB access" mode. Correspondingly, the TTFB Avg values and TTLB Avg values for "EJB-DB access" mode are lower than those of "EJB-ALT-DB access" mode. The Trade2 specification describes that VAJ EJB access beans provide lots of benefits such as "Reduced complexity of EJB access through simplified client APIs; Transparent caching of EJB homes and initial Context objects; Reduced remote calls by making fine grain setter/getter methods local and providing a single refresh/commit remote call ". This experimental result further indicates the performance gains through the use of VAJ EJB access beans.
- In each run time mode, the DB-Light workload mix shows better performance result than DB-Heavy workload mix. These set of results

illustrate that the intensity of database operations affect the performance of a system. Hence, in order to derive accurate system performance prediction, it is important to analyse business requirement and usage scenarios, and to determine the right mix of database intensive operations.

3.2 Sun's ECperf Test Setup and Sample Results

3.2.1 Test Setup

We must use ECperf 's drivers, which are in the test driver package provided as part of the ECperf kit. The drivers directly communicate with the web application server via standard protocol such as RMI/JRMP, RMI/IIOP etc. The Supplier Emulator emulates the process of sending and receiving orders to/from suppliers. The Driver and the Supplier Emulator do not belong to the System Under Test (SUT) (see Figure 7).

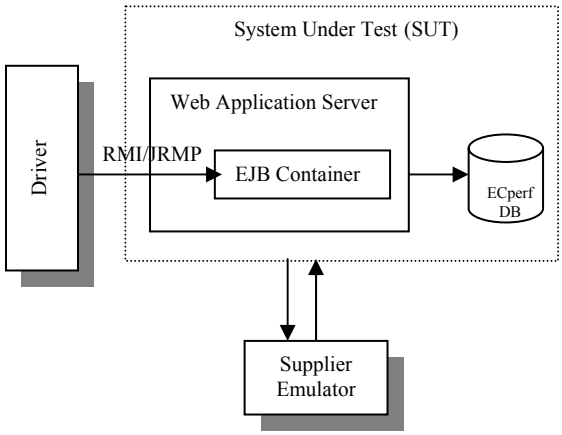

Figure 7: ECperf Test Setup

3.2.2 Sample Results

The following are results obtained with the Injection Rate Ir (ECperf specification 2001) set to be 38. The Ir value is a factor affecting the throughput of ECperf benchmark and the cardinality (in rows) for customer and item related tables. In this run, the ECperf Metric is 3958.13 BBops/min (footnote[2]). Figure 8 – Figure 13 further illustrated the result profiles for this test.

[2] BBops/min is defined to be the total number of business transations completed in the customer domain, added to the total number of work orders completed in the manufacturing domain, normalized per minute (ECperf specification 2001).

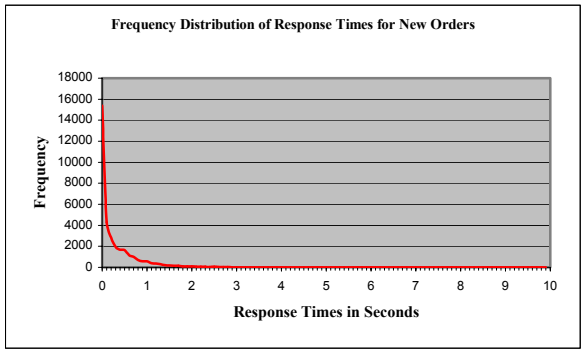

Figure 8: Frequency Distribution of Response Times for New Orders

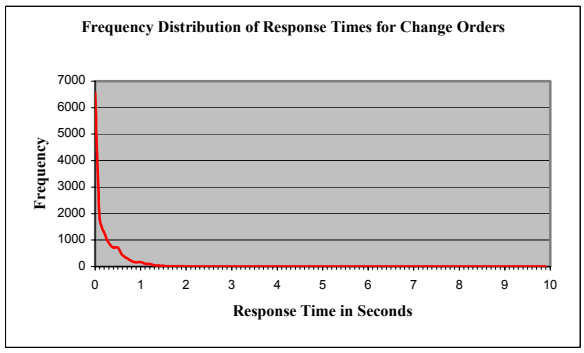

Figure 9: Frequency Distribution of Response Times for Change Orders

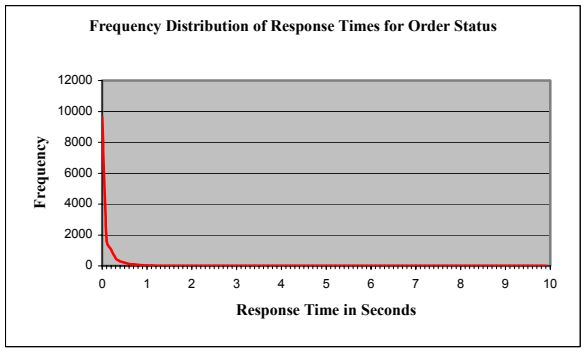

Figure 10: Frequency Distribution of Response Times for Order Status

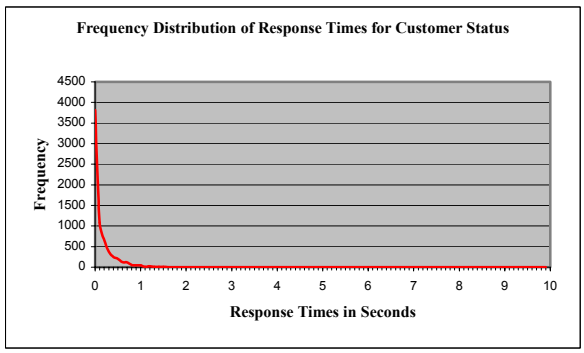

Figure 11: Frequency Distribution of Response Times for Customer Status

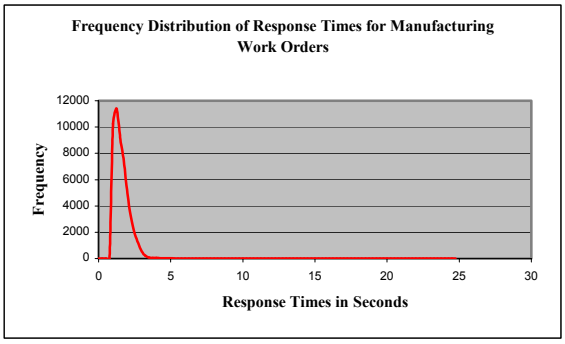

Figure 12: Frequency Distribution of Response Times for Manufacturing Work Orders

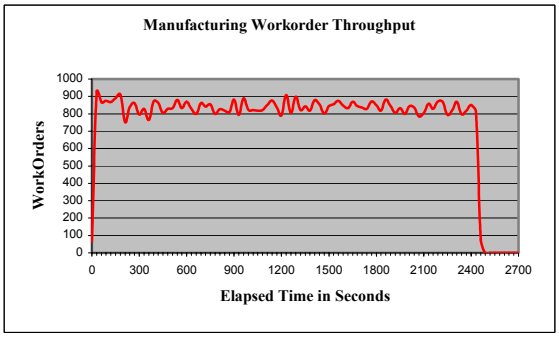

Figure 13: Manufacturing Workorder Throughput

We present some sample statistical data result here to illustrate the sort of performance results obtainable using ECperf. This set of results passed the conditions set by the ECperf specification (ECperf specification 2001). One should note that the performance results presented here is not the highest throughput possible for the WebSphere Application Server on our given test platform. There are numerous tuning (e.g. increasing the Ir value, optimally tuning the application server configurations including systems threads sizes, cache sizes, JVM settings etc) that can be performed in order to get more performance out of a fixed hardware setup and a given J2EE application server. This point cannot be overlooked in examining and comparing J2EE application server benchmark results.

4 Comparison

4.1 Openness

Both Trade2 and ECperf are capable of being deployed on J2EE compliant application servers.

Trade2 : Each of the Trade2 components is written to open web and Java Enterprise APIs, making the Trade2 application portable across J2EE compliant application servers.

ECperf: The ECperf benchmark is an industry standard benchmark for testing J2EE platform-based servers. The wide acceptance and use of this benchmark in testing numerous J2EE application servers indicates its openness (The Server Side Forum 2002)

4.2 Test Driver

Trade2: We must use a web load generator, which is not provided by IBM's Trade2, to drive the Trade2 benchmark (see Figure 3).

ECperf: ECperf kit provides us with test drivers, which directly communicate with the web application server via standard protocols such as RMI/JRMP, RMI/IIOP etc. The Supplier Emulator emulates the process of sending and receiving orders to/from suppliers. The Driver and the Supplier Emulator do not belong to the System Under Test (SUT) (see Figure 7).

Depending on how the test client drives the activities on the server side, the observed throughput of the system will vary. From the perspective of the test result publisher, it is thus important to clearly describe the test driver behaviour when presenting the test results. From the perspective of the test result user, we need to understand the test driver behaviour, and its implication on the test result, in order to accurately interpret test results, and determining whether the test setup and hence result is appropriate for our specific purpose or not.

4.3 Workload Mix

Trade2: Trade2 performance benchmark suite can be configured dynamically to model various e-business environments by adjusting the mix of servlet, session, JSP and EJB operations. It provides three run time modes: "EJB-DB access", "JDBC-DB access" and "EJB-ALT-DB access" and provides five kinds of workload mixes ("Standard", "DB-heavy", "DB-light", "Uniform" and "No Register") to test system. Currently, Trade2 provides the following standard runtime workload mixes (shown in Table 1).

Trade Action	Servlet/EJB	Standard	EJB/JDBC	Uniform
Login	1.5%	3%	5%	10%
Logout	1.5%	3%	5%	10%
Register	2%	4%	5%	10%
Home[3]	25%	20%	20%	10%
Account[4]	5%	7%	8%	10%
Account Update	2%	5%	5%	10%
Portfolio[5]	5%	8%	12%	10%
Quote[6]	50%	40%	30%	10%
Buy	4%	5%	5%	10%
Sell	4%	5%	5%	10%

Table 1: Trade2 Runtime Workload Mixes

[3] Home means personalized home page including current market conditions.

[4] Account means reviewing current user profile information.

[5] Portfolio means viewing users current holdings.

[6] Quota means viewing a current security quote.

ECperf: The ECperf transactions selected by the OrderEntry Driver should meet the following mix requirement which must be within 5% of the target mix for each type of transaction in actual mix achieved in the benchmark (Shown as Table2).

Transaction Type	Percent Mix
NewOrder	50%
getOrderStatus	20%
changeOrder	20%
getCustStatus	10%

Table 2: ECperf Mix Requirements

The mix above is update intensive. This is the purpose of ECperf which is designed to test the transaction handling capabilities of EJB containers. However, most e-business web sites are not update intensive. In fact, studies show that typical web applications perform 85% read only operations/requests, 15% update operations (Transaction Processing Performance Council 2002). This means that most systems' requirement will be less stringent than the one imposed by ECperf.

ECperf dictates a common test scenario for all application servers. The aim is to evaluate the EJB container's performance. ECperf has 4 entity beans and 2 session beans in Customer Domain, 7 entity beans and 2 session beans in Manufacturing Domain, 5 entity beans and 2 session beans in Supplier Domain and 3 entity beans in Corporate domain. These beans are not used all the time, depending on the particular task, different beans are used at different times. The EJB container cannot afford to maintain large number of bean instances in the pool/cache, it has to passivate/activate beans while keeping some active instances in the memory. Hence, the features of the container, such as memory management, connection pooling, passivation/activation and caching etc, contribute towards the final container and server performance. ECperf does not test those features in isolation. It tests the overall performance of the EJB container. Each application server tested using ECperf needs to be optimally "tuned", which is highly subjected to variations. Different application servers may have different configuration parameter settings in order to achieve the best ECperf result for that product. The EJB container that has better memory management, connection pooling etc will give a better performance.

4.4 Database Model

Trade2: Trade2 has 5 database tables which are TradeAccountBean, TradeHoldingBean, TradeProfileBean, TradeQuoteBean and TradeRegistryBean tables. Each table is populated with 500 records by default.

ECperf: ECperf has 4 separate databases with multiple tables. The four databases are Corporate databases which include Customer, Parts, Supplier, Site, Rule, Discount table, Order database which include Customer, Orders, OrderLine, Item tables, Manufacturing database which include Parts, BOM, WorkOrder, LargeOrder, Inventory tables and Supplier database which include Supplier, Site, Component, PurchaseOrder, PurchaseOrderLine, SupplierComponents tables. The cardinality (in rows) of customer and item related tables is related to the Orders Injection Rate Ir (ECperf specification 2001) which is the factor affecting the throughput of ECperf benchmark. We must use a load program provided by the ECperf kit to populate these databases.

The size of database tables, number of database table accesses/joins/search operation for each business transaction, size of database records, etc all affect the performance of the overall system. Hence, the benchmark user needs to determine what will closely represent their business scenario, then use a benchmark that allows for some flexibility in configuring database table parameters.

4.5 Neutrality

Trade2: Trade2 was developed by the IBM WebSphere Performance team. There are VAJ access beans included in Trade2 benchmark (see Figure 1). The VAJ EJB access beans provide lots of benefits which includes performance improvements. This has been shown in our test (see Figure 4 and Figure 6). However, the VAJ access bean developed by IBM cannot be used by other J2EE application servers. The VAJ access bean is a proprietary feature provide by IBM. Hence, a more fair comparison between different J2EE application servers would be not to use any proprietary optimization features such as the VAJ EJB access bean.

ECperf: The ECperf benchmark was developed by Sun Microsystems with the assistance of the Java community to objectively measure the performance of J2EE application servers and containers. ECperf specification does not describe the use of any proprietary J2EE application server features. ECperf has gained acceptance in the J2EE community as an industry standard benchmark (The Server Side Forum 2002), however, one should still be aware of the multitude of configuration options possible, that can be used and misused to present biased test results, based on either proprietary J2EE application server features.

4.6 Real World and Complexity

Both Trade2 and ECperf model real world use case and activities. Trade2 models an online brokerage firm and it can be configured dynamically to model various e-business environments. ECperf models a more complete business chain: Order processing representing retail e-commerce; Just In Time manufacturing relates to workflow; Supplier management represents business to business e-commerce; Corporate functions represents back office functions. Moreover, ECperf can measure both centralized and distributed environments. The ECperf application is more complex than Trade2 and is more similar to enterprise wide IT systems.

On the other hand, the ECperf stresses the update operations, which does not necessarily represent the typical transaction mix of a typical e-business system.

4.7 Results Reporting

Trade2: Trade2 did not create guidelines on submitting and reporting results. Its results rely on the performance metrics or counters in the web test load generator and the run time mode users choose. Hence, the benchmark results are malleable, at least in terms of presentation, and can be susceptible to being misrepresented and misinterpreted. An example of this is: the WebLogic application server and Websphere application server's performance results derived by IBM and BEA respectively using Trade2 benchmark are released in (WebSphere Summit 2002) and (BEA Systems-Products –BEA Weblogic Server 2002),. The results in (WebSphere Summit 2002) show that the performance of the WebSphere application server is better than that of WebLogic application server, but the results in (BEA Systems-Products –BEA Weblogic Server 2002) show that the performance of the WebLogic application server is much better than that of WebSphere application server.

ECperf: ECperf prescribes a detailed method and format for testing and submitting results. This prescription prevents test result misrepresentation and misinterpretation. ECperf defined the BBops/min as the primary performance metric. ECperf also includes a price-performance metric defined as price/BBops/min (the total price of the SUT divided by the reported BBops/min).

5 Conclusion

This paper provides insights into performance benchmarking of J2EE technologies via critically evaluating IBM's Trade2 benchmark application with Sun's ECperf. Both Trade2 and ECperf can be deployed on J2EE compliant application servers. Trade2 provides five different kinds of workload mixes for users to model different runtime environments while ECperf transaction mix requirement is update intensive which is designed to test the transaction handling capabilities of EJB containers. Both Trade2 and ECperf model real world activities, however, with somewhat unrealistic transaction mix percentages ECperf is more complex than Trade2. Trade2 did not create guidelines on submitting and reporting results, which may cause benchmark results to be misrepresented and misinterpreted. ECperf has a detailed specification for testing and submitting results, which prevents such misrepresentation.

From the users' perspective, choosing ECperf or Trade2 as their benchmark to select an application server product depends on their actual business requirement and application characteristics. If their actual application characteristics are similar to Trade2, they can choose Trade2 as their benchmark. If their actual application characteristics are similar to ECperf, they can choose ECperf as their benchmark. However, it is anticipated that one-size fits all T-shirts never quite fit everyone very well, hence, users can expect extensive tuning and configuration required to more closely simulate their business scenario.

Lastly, with these insights into the benchmarking applications, we hope to enlighten future benchmark application design research carried out at CSIRO, and anticipate that these insights will inform the users on how to be more discerning when reading published benchmark results.

6 Acknowledgement

This work is carried out as part of Yan Zhang's visiting PhD studentship at CSIRO. We acknowledge CSIRO's Middleware Technology Evaluation (MTE) research group members for the assistance and support throughout this study.

7 References

BEA Systems-Products –BEA Weblogic Server (2002), http://www.bea.com/products/weblogic/server/performance_ibm_details1.shtml, BEA

Chen, S., Gorton I., Liu A., and Liu Y. (2002): Performance Prediction of COTS Component-based Enterprise Applications, *Proc of the 5th ICSE Workshop on Component-Based Software Engineering Benchmarks for Predictable Assembly*, Orlando, Florida,USA. http://www.sei.cmu.edu/pacc/CBSE5/CBSE5-Proceedings.html

ECperf specification 2001, http://java.sun.com/j2ee/ECperf, Sun Microsystems.

Gorton I. and Liu A. (2002): Software Component Quality Assessment in Practice: Successes and Practical Impediments, *Proceedings of International Conference on Software Engineering (ICSE 2002)*, pp.609-612, IEEE Computer Society Press.

IBM WebSphere Performance Benchmark Sample (2002), http://www-3.ibm.com/software/webservers/appserv/wpbs_download.html , IBM.

Microsoft web stress tool (2002), http://www.microsoft.com/technet/treeview/default.asp?url=/TechNet/itsolutions/intranet/downloads/webstres.asp?frame=true, Microsoft.

The Server Side Forum (2002), http://www.TheServerSide.com

Transaction Processing Performance Council (2002), http://www.tpc.org/

WebSphere Summit 2002, IBM WebSphere software.