Eight Reasons ECperf™ is the Right Way to Evaluate J2EE™ Performance

An Overview of the ECperf Benchmark

October 2001

Salil Deshpande
The Middleware Company
salil@middleware-company.com

Bruce Martin
The Middleware Company
bruce@middleware-company.com

Shanti Subramanyam
Sun Microsystems
shanti.subramanyam@sun.com

The Middleware Company
PO Box 80049 Austin, TX 78708-0049
Austin, TX USA
1 (877) 866-5282
http://www.middleware-company.com

901 San Antonio Road
Palo Alto, CA 94303 USA
http://www.sun.com/
http://java.sun.com/
http://java.sun.com/j2ee/ecperf/

Table of Contents

TABLE OF CONTENTS **2**

1 INTRODUCTION **3**

2 WHY ECPERF? **3**

SUMMARY 6

3 ECPERF EXPLAINED **7**

3.1 THE BENCHMARK APPLICATION 7
3.1.1 The Customer Domain *8*
3.1.2 The Manufacturing Domain *8*
3.1.3 The Supplier Domain *9*
3.1.4 The Corporate Domain *9*
3.1.5 Drivers and Supplier Emulation *9*
3.2 THE SPECIFICATION 9
3.3 WHAT ECPERF IS NOT 9

4 MEASURING AND REPORTING SYSTEM PERFORMANCE **10**

4.1 CHARACTERIZING THE SYSTEM BEING TESTED 10
4.2 CHARACTERIZING THE SYSTEM'S PERFORMANCE 10
4.3 PRICING THE SYSTEM UNDER TEST 10
4.4 DISCLOSING THE RESULTS 10
4.5 PROCEDURES FOR SUBMITTING RESULTS 11

5 ECPERF RESULTS **12**

5.1 ECPERF SUMMARY REPORT 12
5.2 ORDERS SUMMARY REPORT 14
5.3 MANUFACTURING SUMMARY REPORT 15

6 CONCLUSIONS **15**

7 REFERENCES **15**

8 ABOUT THE AUTHORS **16**

1 Introduction

Since its introduction over three years ago, Enterprise JavaBeans (EJB)™ technology has maintained unprecedented momentum among platform providers and enterprise development teams alike. This is due to fact that the EJB server-side component model simplifies development of middleware components that are transactional, scalable, and portable. J2EE application servers supporting EJB technology reduce the complexity of developing middleware by providing automatic support for middleware services such as transactions, security, database connectivity, and more. To accomplish this, J2EE application server products assume all the burden of this complexity, and are gradually becoming the operating system on which all future server-side applications will be built.

The increasing complexity of J2EE application server products therefore has made their performance evaluation and benchmarking somewhat of a black art. Evaluating J2EE products has required a lot of effort and can be very subjective. Some evaluation criteria, such as quality of vendor support and company viability, are necessarily subjective. On the other hand, other criteria, such as performance, scalability, standards conformance and correctness of the implementation need not be subjective if carefully crafted, realistic benchmarks and well defined procedures for running those benchmarks are available.

This paper describes in detail the most objective method devised to date for evaluating the performance of J2EE application servers: *ECperf*™[1].

ECperf has been developed by the Java Community, through the Java Community Process, and published as a standard benchmark for J2EE Enterprise Application Servers.

Some of the unique aspects of the ECperf benchmark are that ECperf is a real-world application with the right level of complexity, that there are strict guidelines for reporting, reviewing, and publishing results, and that the Total Cost of Ownership (TCO) of the system running the benchmark must be included when reporting results.

Section 2 begins this paper by stating the reasons that make ECperf the right way to evaluate J2EE application servers.

Section 3 provides an overview of the ECperf benchmark application, which not only lends support to the statements made in Section 2, but can also be used as a substitute for reading the 100-page ECperf specification document.

Sections 4 provides further information about ECperf's specified process of measuring and reporting system results, and Section 5 provides results of a system we recently tested, as a sample.

2 Why ECperf?

ECperf is an excellent benchmark for measuring the performance and scalability of J2EE enterprise applications, for the following eight reasons:

Reason # 8	**ECperf is gaining acceptance in the community as an independent and neutral benchmark**

ECperf was created by Sun Microsystems with the assistance of the Java community to objectively measure the performance of J2EE application servers. Practitioners who focus on EJB have been

touting ECperf since its inception. Industry research firms and analysts have been learning more about ECperf and in recent months are recommending it as the only objective way to evaluate J2EE application servers[2].

Reason # 7 ECperf has an application server focus

Other benchmarks do exist[3] to measure the performance of the client tier, web server tier, or the database server, but not the middle tier. ECperf, on the other hand, measures the performance of operations on business objects in the middle tier.

In small to medium sized applications, the middle tier is often unimportant. Today, such applications are constructed as a combination of web presentation technology (e.g., Servlets & JSP) and database technology.

However, the middle tier becomes much more important as the application increases in size and complexity, or in the number of users or transactions it must handle. The center of the application becomes the middle tier – the application server.

The performance and scalability of the application server, therefore, has a major impact on the long term health and cost of ownership of enterprise applications. And a benchmark that focuses on the application server is crucial in judging the quality of the application server.

Reason # 6 ECperf exemplifies modern software development techniques

The ECperf application is designed and implemented as a set of interacting reusable and modular Enterprise JavaBeans (EJB). The application design reflects the state of the art in the design and implementation of distributed applications.

Contrast this to other benchmarks, which are designed only to "make things go fast" or measure certain specific functionality, and are horrible examples of how one would really architect, design, and code a real-world application.

This aspect of ECperf becomes even more important as more and more companies move to basing their applications on EJB. These should use ECperf during their selection process for development and deployment environments.

Reason # 5 ECperf is "real-world" and has the right level of complexity

The primary goal of the ECperf workload is to model the performance and scalability of J2EE enterprise applications as seen and implemented by real customers. Isolated low-level, or "unit" benchmarks, such as tests of client-server round-trip requests, are insufficient as real-world benchmarks, because they test the performance of simple operations but not of complete applications.

The ECperf benchmark has the characteristics of real-world systems. For reasons of interest, scope, and familiarity, ECperf uses *manufacturing, supply chain management, and order/inventory* as the "storyline" of the business problem. This is a meaty, industrial-strength distributed application. It is heavyweight, mission-critical, worldwide, 7x24, and necessitates use of a powerful, scalable infrastructure. Many Fortune 500 companies are interested in this application domain because they base their businesses on such IT systems.

Read Section 3 to learn more details, and Section 3.1 in particular, for a detailed description of the benchmark application.

The ECperf application captures intra-, extra-, and inter-company business processes. The application is easy to understand, straightforward to deploy, and run in reasonable time with excellent repeatability. ECperf thus has the right level of complexity for evaluating how real enterprise applications would perform on the J2EE environment being tested.

Reason # 4	**ECperf measures the completeness, compliance, and scalability of a J2EE environment**

The business problem that the ECperf application addresses, necessitates the use of distributed worldwide services and data whose exact location is not known in advance. The application manages persistent data, and needs to be highly available and secure.

In order to meet these requirements, ECperf uses the services required by enterprise applications, as defined by J2EE, including transactional components, distributed ACID transactions, naming services, object (enterprise bean) persistence (both bean-managed and container-managed), and others.

The business problem requires a completely scalable infrastructure. As the size of the modeled business grows, the services, the amount of data, the size of data, the number of users, the number of interactions per user, the number of transactions per time period, and the number of geographically distributed sites also grow. To cope, the server environment must adjust the sizes or numbers of CPUs, memory, application server instances, Java virtual machines, threads, clusters, connection pools, databases, and much more. “Scalability” is a measure of how well an application server makes use of these additional resources.

ECperf measures how effectively a J2EE environment scales. This aspect of ECperf, especially combined with the aspect of Total Cost of Ownership (Reason #1), makes ECperf the obvious way to compare J2EE application servers.

All J2EE compliant application server products should be able to run ECperf. As such, the performance and scaling capabilities of different application servers can be compared.

This ensures that the application server running the ECperf benchmark is indeed a compliant, robust and complete implementation of the J2EE standard.

The J2EE certification test suite, of course, contains numerous unit tests for J2EE compliance. However, ECperf is the first large compliant application developed via the Java Community Process that will stress test much of the J2EE funcationality all at once, in a non-trivial manner.

Beware of the J2EE compliance of application servers that “cannot run” the ECperf application out of the box.

For more information on J2EE compliance, completeness, and compatibility, please visit http://java.sun.com/j2ee/compatibility.html.

Reason # 3	**ECperf prohibits the modification of the application code and the SQL it executes**

With ECperf, the benchmark measures the performance and scalability of an entire application. ECperf prohibits the modification of the application and the SQL the application executes. As

such, the benchmark results from different application servers are comparable. Application server vendors are forced to focus on the tuning of the application server itself and not allowed to play games by tweaking the application or its SQL.

On the other hand, ECperf does not prohibit, but encourages, the modification of deployment aspects of the application. Deployment aspects include the content of deployment descriptors, how the application is packaged, how many containers, machines, Java virtual machines, clusters are in the deployed system, how beans are allocated to containers, etc. This provides the flexibility needed to get the best run time performance and scalability out of an application server.

Reason # 2	**ECperf specifies guidelines on how to submit, report, and publish results**

Many benchmarks, and their results, are malleable and can be susceptible to being twisted into whatever an application server vendor wants to demonstrate. This is especially true of "benchmarks" invented by application server vendors, but can also be true of independent benchmarks.

To avoid this problem, the ECperf Expert Community has created strict guidelines on submitting and reporting results, which prevent such twisting. The guidelines ensure that the results of the benchmark are accurate, and can be verified and duplicated given the appropriate software and documentation. These guidelines are described in sections 4.4 and 4.5.

Reason # 1	**ECperf helps you estimate the Total Cost of Ownership (TCO) of an application over time**

An ECperf test produces a price-performance metric, the price per business operation per minute. A complete system configuration is priced, the benchmark is run and the metric is calculated by dividing the total system price by the number of business operations per minute.

The cost of the system must be reported with the results.

The cost of the system includes:

- the hardware cost (application server & database server machines)
- any operating system cost
- and the cost of the application server and database server license(s).

This aspect of ECperf makes the results of the benchmarks across application server products, operating systems, and hardware, especially accurate and useful.

As you scale the tested system to meet hypothesized increased demand, you reprice the tested system, rerun the benchmark and recompute the price-performance metric. This estimates how much the application will cost over time as your needs grow.

Summary

This section outlined the reasons for the appropriateness of ECperf that, in our experience, are applicable to most environments. Not all of these reasons may apply to yours; however, most will, or should.

To provide support to the statements made in this section, the next section provides an overview of the ECperf benchmark application.

3 ECperf Explained

ECperf is an Enterprise JavaBeans (EJB) benchmark meant to measure the scalability and performance of J2EE servers and containers. It consists of:

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

3.1 The Benchmark Application

The benchmark application is meant to be a realistic enterprise application for a global corporation. The intention is for it to be simple enough to understand and measure, yet complex enough to be realistic.

The benchmark application is structured as Enterprise JavaBean™ components. It is intended to run on any J2EE compliant product.

The application is delivered as a complete "kit" that includes the Enterprise JavaBeans, a set of Java ServerPages™ for single user/interactive testing, schema scripts and load programs to create and load the database(s) and driver programs to implement the run rules and simulate the client load. All Java source code and make files are included.

The ECperf application implements manufacturing, supply chain management, and order/inventory for a large global enterprise. As shown in Figure 1, the application consists of four application domains: *manufacturing*, *supplier & service provider*, *customer*, and *corporate*. It is assumed that each domain has separate databases and applications. Most likely, they are implemented on separate computing hardware. Since large corporations have often merged with other corporations, the various application domains may themselves be very heterogeneous.

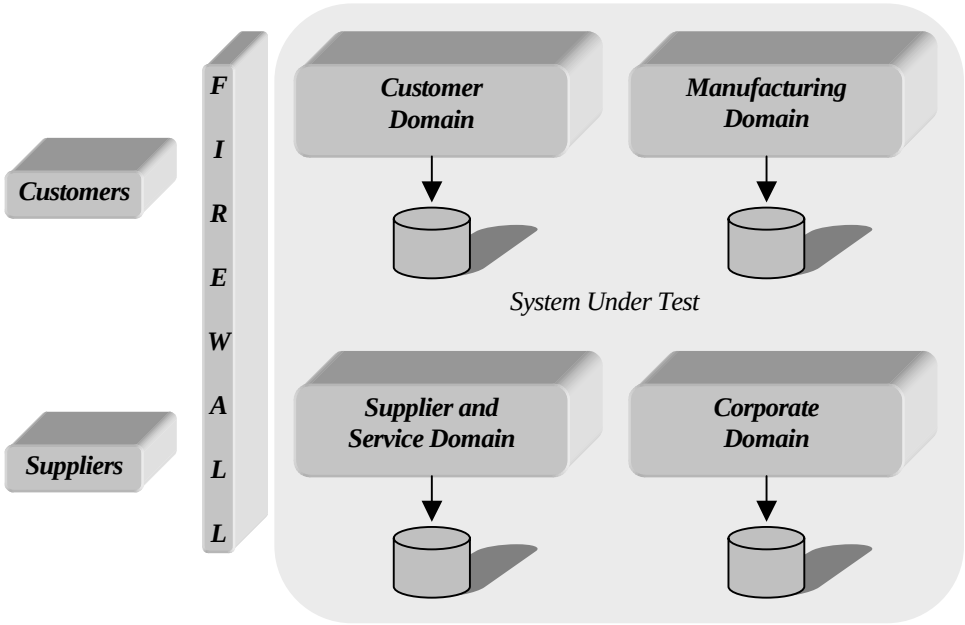

Figure 1 – The ECPerf Application

There are producer-consumer relationships between domains in the company and to outside suppliers and customers as well.

Customers contact the business through any number of methods, including through the web, over the telephone or through a salesperson. All of these methods map into this web access scheme because the customer service representative or salesperson themselves use the web interface. All of the worldwide offices and plants make frequent access to data held in the other offices or plants, and must, at times, compare/collate/verify their data against that held worldwide.

The company also interacts with completely separate supplier companies. Each supplier has its own independent set of computing resources.

3.1.1 The Customer Domain

Work in the customer domain is OLTP in nature. An order entry application supports adding new orders, changing an existing order and retrieving status of a particular order or all orders of a particular customer.

Orders are placed by individual customers as well as by distributors. The difference between the two is in the quantity of items ordered. Approximately 57% of work on the system are large orders from distributors and 43% are regular orders from individual customers.

A credit check is performed on the customer by sending a request to the corporate domain. Various discounts are applied to the order depending on whether the customer is a distributor, repeat or first-time customer, etc.

Existing orders may be changed. The quantities of ordered items may be changed or the order may be cancelled in its entirety. Orders that have already shipped or have entered the shipping process cannot be cancelled.

3.1.2 The Manufacturing Domain

This domain models the activity of production lines in a manufacturing plant. Planned production lines run on schedule and produce a pre-defined number of widgets. On the other hand, the other production lines run only when a large order is received from a customer such as a distributor.

Manufacturing begins when a work order enters the system. Each work order is for a specific quantity of a particular type of widget. The planned line work orders are typically created as a result of a forecasting application. The large order line work orders are generated as a result of customer orders. When a work order is created, the Bill of Materials (BOM) for the corresponding type of widget is retrieved and the required parts are taken out of inventory. As the widgets move through the assembly line, the work order status is updated to reflect progress. Once a work order is complete, it is marked as complete and inventory updated.

As inventory of parts is depleted, suppliers need to be located and purchase orders need to be sent out. This is done by contacting the supplier domain.

3.1.3 The Supplier Domain

The supplier domain is responsible for interactions with suppliers. It determines which supplier to choose based on the parts that need to be ordered, the time in which they are required and the price quoted by suppliers. The company sends a purchase order to the selected supplier. The purchase order includes the quantity of various parts being purchased, the shipping address and the required delivery date. When parts are received from the supplier, the supplier domain sends a message to the manufacturing domain to update inventory.

3.1.4 The Corporate Domain

The corporate domain manages the global list of customers, parts and suppliers. Credit information is kept in a database in the corporate domain. This is to provide maximal security and privacy.

For each new order, the customer domain requests the corporate domain to report on the credit worthiness of the customer. Customer discounts are also computed in the corporate domain for each new order or whenever an order is changed..

3.1.5 Drivers and Supplier Emulation

Obviously, the benchmark cannot be driven by real customers placing actual orders, by real manufacturing facilities producing products and by real suppliers providing goods and services. Instead, ECperf defines and provides:

- an *order entry driver* that repeatedly executes business transactions in the customer domain,
- a *manufacturing driver* that drives the planned and large order lines and
- a *supplier emulator*, implemented as a Java servlet in a separate web container to emulate the process of sending and receiving orders to/from suppliers. The emulator and the Supplier Domain communicate by exchanging XML documents.

The drivers are also responsible for recording all relevant statistics and for printing the reports from the run.

3.2 The Specification

The ECperf specification provides detailed descriptions of the application, the object model embodied in the application, the data model, the workload, the rules for running and scaling the workload, the system being tested, the price/performance metrics and the rules for disclosing benchmark results. Although we provide an overview of these, the specification[1] should be read for the definitive discussion of ECperf.

3.3 What ECperf is not

ECperf is not designed to test DBMS performance and scalability. These are adequately measured by other standard workloads such as *TPC-C, TPC-D and TPC-W[2]*.

ECperf is also not a benchmark of client side GUI performance. It is not a test of web browser performance. It does not test “hits” of a web server. ECperf does not test JSP/servlet performance.

4 Measuring and Reporting System Performance

Besides defining the benchmark application, ECperf defines how to characterize the system being tested, its performance and its price-performance ratio. ECperf also defines how to disclose the results of running the benchmark. The specification is the definitive source for this information. We summarize it here.

4.1 Characterizing the system being tested

ECperf defines the *system under test*, or SUT. The SUT comprises all hardware and software components being tested including:

- the application servers
- the database servers
- the hardware and software required to support the workload and databases
- all networking components between host machines and all network interfaces to the SUT
- all components providing load balancing within the SUT
- any software and hardware used to influence the flow of traffic beyond basic IP routing and switching

The supplier emulator and the drivers are not part of the SUT. They cannot be on machines that are in the SUT. The intent is that they communicate with the SUT over the network.

In figure 1 a distributed system under test is graphically represented as the shaded box. The SUT consists of the application servers for each domain, the database servers, the computers running these and the networking components connecting them.

4.2 Characterizing the system's performance

The term **BBops/min** is the primary ECperf metric and denotes the average number of successful Benchmark Business OPerationS per minute completed during the measurement interval. A BBop/min is defined to be the total number of business transactions completed in the customer domain, added to the total number of work orders completed in the manufacturing domain, normalized per minute.

4.3 Pricing the system under test

ECperf includes a price-performance metric defined as price/BBops/min. This metric is the total price of the SUT divided by the reported BBops/minute.

The intent of the pricing rules is to price the tested system at the full price a customer would pay. All pricing sources and effective dates of the prices must be included. The disclosed pricing must reflect the level of detail a customer would see on an itemized billing. Clause 6 of the ECperf specification provides more details.

4.4 Disclosing the results

A full disclosure report is required in order for results to be considered compliant with the ECperf benchmark specification. The intent of this disclosure is for anyone to be able to replicate the results of this benchmark given the appropriate documentation and software. Clause 7 of the ECperf specification is the definitive source for the contents of the report. Table 1 summarizes the contents of the report.

Table 1: Information that is required to be submitted with an ECperf results report

Report Must Disclose	*Definition*
Benchmark sponsor	A statement identifying the benchmark sponsors and other participating companies.
Configurations	Diagrams of both measured and priced configurations.
Persistence type	The type of persistence used (CMP, BMP or mixed mode).
Deployment descriptors	All deployment descriptors used.
Software products	All commercially available software products used.
Output	The entire output directory from the reproducibility run.
Database setup	All database table definition statements, statements used to set up the databases and the order injection rate used to load the databases.
Client systems	The number and types of client systems used, along with the number and types of processors, memory and network configuration
Priced system	A detailed list of hardware and software used in the priced system.
Sophisticated IP routing	Any software/hardware used to influence the flow of network traffic beyond basic IP routing and switching.
Load balancing	Details of any load-balancing functions.
Driver input	The input parameters to the driver.
Network bandwidth	The bandwidth of the network(s) used in the tested/priced configuration.
Communication protocol	The protocol used by the Driver to communicate with the SUT (e.g RMI/IIOP).
Transaction ACID tests	The scripts/programs used to run the ECperf tests of the transaction ACID properties.
Performance results	■ The BBops/minute from the reproducibility run. ■ A graph of the frequency distribution of response times for all the transactions. ■ A graph of the work order throughput versus elapsed time.

4.5 Procedures for submitting results

When a test sponsor decides to publish a ECperf benchmark result, he must generate the full disclosure report and submit it to the ECperf Review Committee by electronic mail. The ECperf Review Committee has a minimum of two weeks to review the result and will vote on accepting it at their regularly scheduled meeting after the two week period.

During the review period, the members of the committee will hold all review material in strict confidentiality. Within a member company this information should be shared with only those people necessary to provide a thorough review of the result.

At the end of the review, if there are no open issues related to a given submission, the result is approved and the full disclosure is posted on the ECperf results website. Once posted, the information contained in the disclosure is considered public. If there are open issues for a given submission at the end of its review cycle that have not and can not be resolved to the satisfaction of the majority of the committee members, the committee votes that the submission is not compliant and should not be published. The sponsor is informed of the problems with his result including all relevant clauses of the specification that are not met.

Clause 8 of the ECperf specification provides more details.

5 ECperf Results

ECperf produces several result files. Examples of these result files are shown in figures 2, 3 and 4. This section explains the results.

5.1 ECperf Summary Report

The ECperf Summary reports the parameters under which the benchmark was run and the benchmark business operation result. Table 2 defines the most frequently modified parameters.

Table 2: Parameters that are most frequently modified while running the ECperf benchmark

Parameter	*Definition*
`runOrderEntry`	Indicates if the OrderEntry driver is to be run.
`runMfg`	Indicates if the Manufacturing driver is to be run.
`txRate`	Indicates the transaction injection rate. It determines how many client threads are started and the rate at which they access the server. In particular, 5*txRate threads are started to do order entry transactions and 3*txRate threads are started to do manufacturing transactions for planned lines.
`rampUp`	Sets the number of seconds to wait before collecting statistics. The benchmark assumes the server is in a steady state after *rampUp* seconds.
`rampDown`	Sets the number of seconds after the measurement interval that the benchmark must continue to run
`stdyState`	Sets the number of seconds for the measurement interval.
`triggerTime`	Sets the number of seconds it takes for all the client threads to start up. This depends on the number of threads being started and the JVM being used. This value is set by trial and error.

Figure 2 – ECperf Summary Report: run parameters and benchmark business operations

```
                ECPerf Summary Report
                Version : ECperf 1.0 Update 1


Run Parameters :
runOrderEntry = 1
runMfg = 1
txRate = 58
rampUp (in seconds) = 600
rampDown (in seconds) = 300
stdyState (in seconds) = 1800
triggerTime (in seconds) = 90
numOrdersAgents = 1, numMfgAgents = 1
dumpStats = 0
Benchmark Started At : Fri Oct 12 15:31:30 GMT-08:00 2001

Orders Summary report is in : Orders.summary
Orders Detailed report is in : Orders.detail
Orders Transaction Rate : 3460.13 Transactions/min

Manufacturing Summary report is in : Mfg.summary
Manufacturing Detail report is in : Mfg.detail
Manufaturing Rate : 2538.00 WorkOrders/min

ECperf Metric : 5998.13 BBops/min
```

5.2 Orders Summary Report

Figure 3 – ECperf Orders Summary Report

```
               Orders Summary Report
               Version : ECperf 1.0 Update 1

Orders Transaction Rate : 3460.13 Transactions/min

TRANSACTION MIX

Total number of transactions = 103804
TYPE            TX. COUNT    MIX          REQD. MIX.
----            ---------    ---          ----------
NewOrder:       51884        49.98%       50%                PASSED
ChangeOrder:    20811        20.05%       20%                PASSED
OrderStatus:    20701        19.94%       20%                PASSED
CustStatus:     10408        10.03%       10%                PASSED
ECPerf Requirement PASSED

RESPONSE TIMES    AVG.MAX.     90TH%   REQD. 90TH%
--------------    ----       ----    -----   -----------
NewOrder           0.612      6.232   1.300   2
ChgOrder           0.480      4.151   1.000   2
OrderStatus        0.180      2.518   0.500   2
CustStatus         0.518      5.950   1.400   2
ECPerf Requirement for 90% Response Time PASSED
ECPerf Requirement for Avg. Response Time PASSED


CYCLE TIMES     TARGETED AVG. ACTUAL AVG. MIN.   MAX.
-----------     ------------- ----------- ----   ----
NewOrder        4.949          5.012        0.000  25.000    PASSED
ChgOrder        4.944          4.987        0.000  25.000    PASSED
OrderStatus    4.966          4.974        0.000  25.000    PASSED
CustStatus     4.969          5.034        0.000  25.000    PASSED

MISC. STATISTICS

Average items per order                   28.658
Widget Ordering Rate                      49562.433/min      PASSED
Percent orders that are Large Orders      10.10              PASSED
Average items per Large order             150.110            PASSED
Largeorder Widget Ordering Rate           26229.233/min      PASSED
Average items per Regular order           15.008             PASSED
Regular Widget Ordering Rate              23333.200/min      PASSED
Percent orders submitted from Cart        50.08              PASSED
Percent ChgOrders that were delete        9.99               PASSED


LITTLE'S LAW VERIFICATION


Number of users = 290
Sum of Avg. RT * TPS for all Tx. Types = 288.448289
```

5.3 Manufacturing Summary Report

Figure 4 – ECperf Manufacturing Summary Report

```
                Mfg Summary Report
                Version : ECperf 1.0 Update 1

Total Number of WorkOrders Processed : 76140
Number of WorkOrders as a result of LargeOrders : 14021
Total WorkOrders Production Rate : 2538.00 WorkOrders/min
LargeOrders Production Rate : 467.37 LargeOrders/min

Total Widget Manufacturing Rate : 46819.63 widgets/min
LargeOrderLine Widget Rate :       23525.87 widgets/min    PASSED
PlannedLines Widget Rate :        23293.77 widgets/min     PASSED

RESPONSE TIMES     AVG.      MAX.       90TH%     REQD. 90TH%
--------------    ----      ----      -----     -----------
                  2.216     14.971    3.000     5
ECPerf Requirement for 90% Response Time Passed
ECPerf Requirement for Avg. Response Time Passed
```

6 Conclusions

ECperf is an excellent way to measure the performance and scalability of J2EE application servers. It defines a real-world enterprise application benchmark of a J2EE application server. The benchmark allows for a fair comparison of application servers. ECperf was not defined to favor any particular product. It was defined by the Java Community Process.

ECperf is very explicit in the procedures for running and reporting benchmarks so that a given run of a benchmark is repeatable by others. ECperf performance results for various application servers and various system configurations are to be submitted to the ECperf Review Committee for review and publication on their web site. When this happens, we will finally be able to fairly compare the performance and scalability of different J2EE application server products.

7 References

1) **ECperf™ Specification**
Version 1.0, Final Release, Sun Microsystems, Inc, http://java.sun.com/j2ee/ecperf/

2) **"Adopt ECperf for J2EE Application Server Benchmarking." Giga Information Group. Idea Byte, September 13, 2001. John Meyer.**
"How does one get an "apples to apples" comparison when benchmarking J2EE application servers? Answer: ECperf. Giga recommends that users adopt the ECperf 1.0 Specification and Kit (which provides the necessary code to run the ECperf benchmark) as the foundation for all application server benchmarking efforts...." Review this article at http://www.gigaweb.com/

3) **TPC Benchmark Standard Specifications**
Transaction Processing Performance Council (TPC) http://www.tpc.org

8 About The Authors

Salil Deshpande is President of The Middleware Company, a worldwide training & consulting company focusing on enterprise software and middleware, and creators of theServerSide.com, the world's largest J2EE and Web Services community. In 1994 Salil founded CustomWare, a consulting and system integration company that focused on CORBA technology, which was acquired by Visigenic Software, which was later acquired by Borland. In 1998, Salil spun out the old CustomWare group from Borland, to form a new company, The New CustomWare Company, which continued to focus on enterprise software, but with a special emphasis on on Java 2 Enterprise Edition (J2EE), Enterprise JavaBeans (EJB), and web services technologies. In 2002, in a move that was praised by the J2EE community, CustomWare merged with The Middleware Company, and thus became the obvious choice for J2EE and Web Services consulting, training, mentoring, skills transfer, and and custom software development. Salil holds a B.S. in Electrical Engineering from Cornell University, and an M.S. in Electrical Engineering / Computer Science (distributed operating systems & programming languages) from Stanford University.

Bruce Martin is Senior Architect at The Middleware Company. Bruce has vast expertise in building scalable multi-tier systems, Java, C++, CORBA, EJB and Java 2 Enterprise Edition (J2EE). He has published numerous papers and spoken at numerous conferences. Bruce leads the Xbeans open-source project that is building a collection of configurable JavaBeans to process XML in a distributed data pipeline. Prior to CustomWare, Bruce held key positions at jGuru, Borland, Sun Microsystems, and Hewlett-Packard Laboratories. Bruce is the principal author of five of the OMG's CORBAServices standards. Bruce received a Ph.D. in Computer Science from the University of California, San Diego in 1988, an MS in Computer Science from University of California, San Diego in 1984 and a BA in Computer Science from the University of California, Berkeley in 1980.

Shanti Subramanyam is a member of the Performance and Availaibility Engineering (PAE) group in Sun Microsystems. She is the Spec Lead for ECperf, a benchmark meant to measure J2EE server performance. She has over ten years of experience in performance analysis and tuning of database products. She has been involved with ECperf and EJB/J2EE performance for over 3 years.