# WebSphere Application Server 6.1 Base Performance

# Table of Contents

## Introduction to the WebSphere Application Server performance tests

### Objectives

The objectives[1] of this project were to gather Linux® end-to-end measurements for a typical transactional workload on WebSphere® Application Server using the Trade application. We gathered performance data from all of the components in the transaction path, from the user accessing the WebSphere Application Server system to the database. The products in the transaction path were: RedHat Enterprise Linux (RHEL), WebSphere Application Server, and IBM DB2® Universal Database™ (DB2) on Linux for IBM System z™. We wanted to show how this set of products, which is needed to run the WebSphere Application Server Trade 6 benchmark on Linux for System z, performs and how to optimize these products in a customer-like environment. We performed release-to-release comparisons, CPU scaling runs, to determine the size of the workload that can be run on a certain amount of CPUs. We also determined how the workload size can grow with an increasing number of CPUs.

This is a continuation of a previous project described at
http://www.**ibm.com**/developerworks/linux/linux390/perf/tuning_pap_websphere.html#wasbp

### Executive summary

We did not see any major performance issues when using the new versions of WebSphere Application Server (6.1.0.11) or RHEL (5.0). We recommend using RHEL 4.5 or higher for WebSphere Application Server workloads. We were not able to obtain RHEL 4.4 data because of performance issues, which are fixed in RHEL 4.5.

For Trade application tuning:

- We showed in a previous paper that the network bandwidth has a major impact, especially the connection between WebSphere Application Server and the database. Therefore, we used for this connection HiperSockets™ and increased the number of I/O buffers for all network connection.
- Our middle class disk I/O subsystem with an ESCON® connected ESS F20 storage server was sufficient to provide the needed disk I/O bandwidth.
- The CPU scalability of this workload was very good. This shows that customers with similar workloads should be able to easily react to growing business by just adding more CPUs instead of installing multiple server environments.

[1] This paper is intended to provide information regarding performance of environments using WebSphere Application Server 6.1. It discusses findings based on configurations that were created and tested under laboratory conditions. These findings may not be realized in all customer environments, and implementation in such environments may require additional steps, configurations, and performance analysis. The information herein is provided "AS IS" with no warranties, express or implied. This information does not constitute a specification or form part of the warranty for any IBM products.

## Summary for the WebSphere Application Server performance tests

After running the WebSphere Application Server performance tests on our test environment, we compiled a summary of our results and recommendations.

Our test results and recommendations are specific to our environment. Parameters useful in our environment might be useful in other environments, but are dependent on application usage and system configuration. You will need to determine what works best for your environment. For our detailed test results information, see Results for the WebSphere Application Server performance tests.

The following are our summary results:

### *Release to release comparisons*

- WebSphere Application Server version 6.1.0.0 versus version 6.1.0.11
  From a performance perspective, WebSphere Application Server version 6.1.0.11 behaves the same as version 6.1.0.0.
- Red Hat distributions
  - We recommend using RHEL 4.5 or higher for WebSphere Application Server workloads. We were not able to obtain RHEL 4.4 data because of performance issues which are fixed in RHEL 4.5.
  - RHEL 4.5 versus RHEL 5.0
    From a performance perspective, the WebSphere Application Server environment on RHEL 4.5 behaves the same as when it is on RHEL 5.0.

### *Tuning variations*

- I/O latencies
  - Disk I/O
    We showed that an application like Trade has a low dependency on disk I/O bandwidth as long as the buffer pools from the database are large enough. Moving the database, which is the only component doing disk I/O, to a RAM disk, which could be considered as the fastest disk I/O device possible, gave only a very slight improvement. Which demonstrates that the disk I/O bandwidth was no bottle neck in our tests.

Note: This was done just for testing purposes. Never use a non-persistent disk device for any component of a database!

- Network
  - We showed in a previous paper that the network connection between the application server and the database has a significant impact on throughput. Therefore, in most cases, we used HiperSockets for our network connections (see http://publib.boulder.ibm.com/infocenter/systems/topic/liaag/WASbaseperformance/publisheddwasperf37.htm#PublishedWASPerf-gen36 for details).
  - Increasing the number of network buffers on the network interfaces improved throughput by 3%.

*CPU scalability*

Scaling the number of CPUs from one to eight gave us very linear throughput and cost scaling, with a scaling factor close to the ideal case. We were able to saturate the WebSphere Application Server CPUs for each scaling point.

## Hardware and software configuration for the WebSphere Application Server performance tests

To perform our WebSphere Application Server tests, we created a customer-like environment. We configured the hardware, software, network, and storage server.

## Hardware and software

The following section details the hardware and software we used for our test runs.

*Server hardware*

Host

Two LPARs on a 16-way IBM System z9® Enterprise Class (z9™ EC), model 2094-S18, configured with:

- LPAR 1 (WebSphere Application Server on Linux)
  - 1 - 8 physical CPUs, dedicated
  - 4 GB central memory
- LPAR 2 (UDB database on Linux)
  - 1 - 8 physical CPUs, dedicated
  - 12 GB central memory
- 2 OSA Express2 Ethernet cards
- 4 dedicated ESCON Express Channels

Network setup

- 1 - 2 client workstations on a 1 Gb Ethernet LAN
- 2 OSA Express2 Ethernet cards on IBM System z, or
- One OSA Express2 Ethernet card and one HiperSockets connection

Storage server setup

- For the RHEL4.5 environment:
  2105-F20, Disk Drive Modules - 18.2 GB each/10 K RPMs
  - 12 ECKD™ mod3s from 1 rank/LCU
  - 4 ESCON paths
- For the RHEL5.0 environment:
  2105-F20, Disk Drive Modules - 18.2 GB each/10 K RPMs (used for the LPAR containing WebSphere)
  - 3 ECKD mod9s from 1 rank/LCU
  - 4 ESCON paths
    2105 Model 800, 72 Gb/10 K RPM DDMs (used for the LPAR containing the DB2 databases)
  - 6 ECKD mod9s from 1 rank/LCU
  - 4 FICON® channels

Note: The usage of ESCON channels and disks from one rank is only suitable for environments with a low disk I/O bandwidth requirement.

### *Server software*

Table 1. Server software used

| Product | Version/Level |
|---|---|
| IBM DB2 9 for Linux UNIX and Windows® | DB2 v9.1.0.2 Fix Pack 2 |
| Red Hat Enterprise Linux (RHEL) | 4.5 |
| Red Hat Enterprise Linux (RHEL) | 5.0 |
| WebSphere Application Server (31-bit) | 6.1.0.0 |
| WebSphere Application Server (31-bit) | 6.1 FixPack 11 |

### *Client hardware*

- 1 IBM eServer™ xSeries® model 6565-2BU with a 667 MHz Pentium III
- 1 xSeries model 6792-MHU with a 1.8 GHz Pentium 4

### *Client Software*

Table 2 shows the client software used.

Table 2. Client software used

| Product | Version/Level |
|---|---|
| xSeries model 6565-2BU and 6792-MHU | |
| Red Hat Enterprise Linux | Release 9 |
| Trade | 6.1 |

### Environment

The environment used for our WebSphere Application Server tests is shown in Figure 1

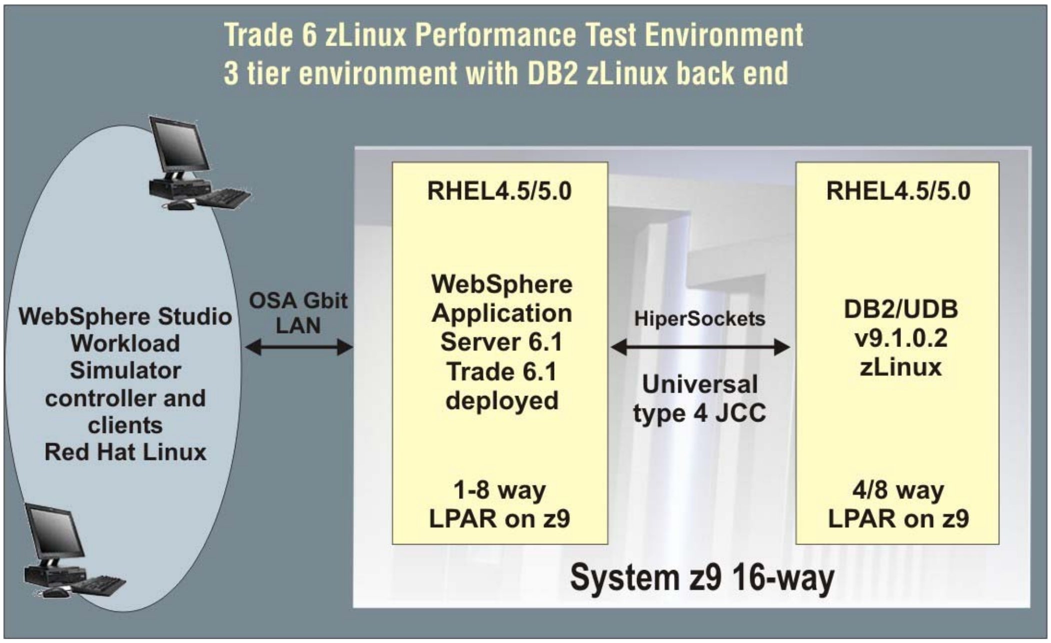


Figure 1: Trade 6 Linux for System z performance test environment with WebSphere Application Server and DB2 on Linux

### Workload description - Trade

Trade is an IBM-developed workload, modeling an electronic brokerage, providing online securities trading.

Trade provides a real-world eBusiness application mix of Servlets, JSPs, EJBs, and JDBC data access, adjustable to emulate various work environments. Trade was deployed under WebSphere 6.1 and DB2 version 9 served as the database backend on Linux on System z.

### Tools description

For our WebSphere Application Server performance tests we used SAR to collect performance statistics data.

#### *SAR*

The SAR program collects reports and saves system activity information. The contents of selected cumulative activity counters in the operating system are written to standard output. The accounting system then uses supplied SAR parameters and writes information based on the given parameters.

#### Instruction sequence

We performed the following steps for each run:

1. Start WebSphere Application Server
2. Load the database
3. Submit the runstats
4. Edit the Trade configuration scripts
5. Warm up the Trade workload for a minimum of five minutes
6. Stop the workload
7. Reset the database
8. Restart the workload
9. Start the SAR data collection
10. Start the monitor data collection
11. Run the workload for approximately seven minutes
12. Stop the workload
13. Stop WebSphere Application Server
14. Generate the SAR report

### System setup for the WebSphere Application Server performance tests

To emulate a customer-like environment we needed to setup our network, the WebSphere Studio Workload Simulator client, the workload, and configure the xpram device for one test case.

### Network setup

Our network setup used an isolated network connection via OSA card for connecting to the clients and a HiperSockets network connection between the WebSphere Application Server and the database (see Figure 2).

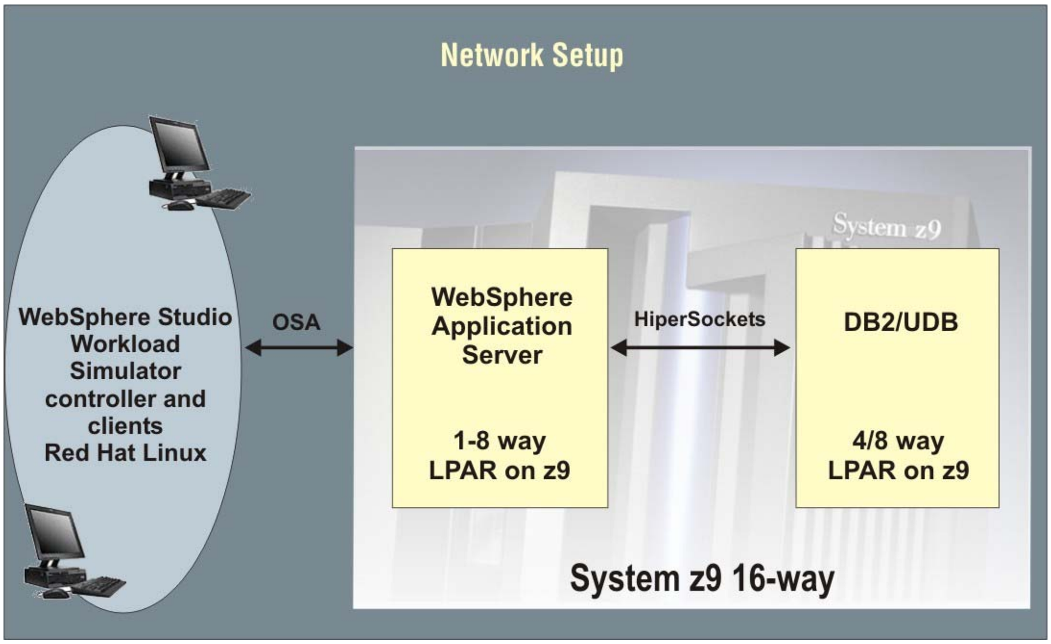


Figure 2. Overview on network setup for the performance test environment

### Changing buffer settings of the network interfaces

In order to perform our WebSphere Application Server performance tests, we needed to change our buffer settings.

For our measurements, we made the following changes to our buffer settings:

- The buffer count for our network connection was changed from 16 to 128.
- The checksumming setting was changed from the default, software checksumming, to hardware checksumming.

Note: For HiperSockets devices, hardware checksumming has no impact, because it does not provide the necessary hardware. Even when specified, checksumming will be done in software.

There is more than one way to change your buffer settings. Two methods are described below.

**Method 1**
We used the following steps to make our changes:

- Take the devices down
  ifdown eth1
- Take the devices offline
  echo 0 > /sys/bus/ccwgroup.drivers/qeth/0.0.0700/online
- Change buffer count
  echo 128 > /sys/bus/ccwgroup/drivers/qeth/0.0.0700/buffer_count
- Change checksumming
  echo hw_checksumming > /sys/bus/ccwgroup/drivers/qeth/0.0.0700/checksumming
- Put devices back online
  echo 1 > /sys/bus/ccwgroup/drivers/qeth/0.0.0700/online
- Restart the network
  service network restart

**Method 2**
The hardware configuration for qeth is found in the file "/etc/sysconfig/hardware/hwcfg-qeth-bus-ccw-0.0.nnnn ". Where "nnnn" is the device number of the appropriate network device.

We added the following line to change the number of buffers from 16, the default, to 128.

QETH_OPTIONS='checksumming=hw_checksumming buffer_count=128'

## WebSphere Studio Workload Simulator client setup

We used WebSphere Studio Workload Simulator clients to drive our performance tests. In many cases we scaled the number of clients and engine setup, for our workload generation, to identify the scenario with the highest throughput. In any case, the WebSphere Studio Workload Simulator clients used were divided equally between two WebSphere Studio Workload Simulator engines.

### Configuring the xpram device on RHEL

For one performance test, we configured the xpram device.

The steps we used to configure the xpram device on RHEL were:

1. Create a directory to mount the xpram device
   mkdir /db2logs_xpram
2. Change the permissions of the directory so that data can be written to it
   chmod 777 /db2logs_xpram
3. Edit the file /etc/rc.local and add the following lines
   modprobe xpram devs=1
   mkfs.ext3 /dev/slram0
   sleep 1
   mount /dev/slram0 /db2logs_xpram
4. Run /etc/rc.local
5. Issue the df command to check if the xpram device was mounted correctly

| Filesystem | 1K-blocks | Used | Available | Use% | Mounted on |
|---|---|---|---|---|---|
| /dev/slram0 | 2064204 | 35880 | 1923468 | 2% | /db2logs xpram |
| /dev/slram1 | 2064204 | 35880 | 1923468 | 2% | /db2data xpram |

For the data files, we configured slram1 accordingly.

*Collecting the output*

This section details the steps we took to gather the output for our test runs.

### Linux statistics

The following data was collected for each measurement:

- sadc/sar
  Generates CPU load, disk and network throughput.

### Trade output

Below is a sample of the output produced by WebSphere Studio Workload Simulator. The key items that were used for throughput are Page element throughput, Transactions throughput, and HTTP average page element response time.

Cumulative statistics are printed every five minutes and at the end of the WebSphere Studio Workload Simulator execution time interval, which was set to 1200 seconds for each of our measurements.

```
====================Cumulative Statistics=======================
IWL0038I Run time = 00:15:02
IWL0007I Clients completed = 0/30
IWL0059I Page elements = 629404
IWL0060I Page element throughput = 697.325 /s
IWL0059I Transactions = 521099
IWL0060I Transaction throughput = 577.332 /s
IWL0059I Network I/O errors = 0
IWL0059I Web server errors = 0
IWL0059I Num of pages retrieved = 629404
IWL0060I Page throughput = 697.325 /s
IWL0060I HTTP data read = 5657.374 MB
IWL0060I HTTP data written = 240.236 MB
IWL0060I HTTP avg. page element response time = 0.041
IWL0060I HTTP avg. page element response time = 0.041
        (with all clients concurrently running)
======== S h a r e d   V a r i a b l e   R e p o r t ========
int curClient = 358
==== E n d   S h a r e d   V a r i a b l e   R e p o r t ====
```

## Results for the WebSphere Application Server performance tests

After performing our WebSphere Application Server performance tests, we charted and interpreted the results, and created recommendations.

The terms ETR and ITR are used throughout this section. These terms are defined as follows:

**ETR**
External Throughput Rate – the actual throughput measured during a test run. For our purposes, it is always expressed as the number of transactions per second measured during the test.

**ITR**
Internal Throughput Rate – the highest possible throughput that could be realized if the test was run at 100% CPU utilization. The ITR is always expressed as the number of transactions per second.

Note: When we state transactions per second we mean page elements per second, as reported by Trade.

The ETR and actual % CPU usage are used to formulate the ITR. The equation used was:

ITR = ETR/% CPU * 100%

We normalized the values for ETR and ITR such that the first ETR value in a sequence was set to 100%.

The tests we performed were:

- WebSphere Application Server release to release comparison
- RHEL 4.5 Trade tuning variations
  - Disk tuning variation: impact of 2105 - F20 disk latencies
  - Tuning variation: Linux network settings
- RedHat RHEL release to release comparison
  - DB2 database on Linux
- RHEL 5.0 scalability

## WebSphere Application Server release to release comparison

The purpose of this test was to determine which level of WebSphere Application Server would give us the best throughput performance. We wanted to determine which code level (6.1.0.0 or 6.1.0.11) would be the best to use for the rest of our tests.

### Test case description

In this test we ran the Trade 6 (Trade) workload in our standard 3 tier environment using RHEL 4.5 for WebSphere and DB2 for Linux. The difference between the two runs was the that one run used WebSphere Application Server version 6.1.0.0 and the other run used WebSphere Application Server version 6.1.0.11.

Table 3. Release to release comparison test cases

| | WebSphere Application Server | | | Trade Results | | | | DB2 Database | |
|---|---|---|---|---|---|---|---|---|---|
| Distro | Code Level | #CPU | %CPU | #Users | Resp | %ETR | %ITR | #CPU | %CPU |
| RHEL 4.5 | **6.1.0.0** | 4 | 99.36 | 60 | 34 ms | 100.00 | 100.65 | 4 | 50.03 |
| RHEL 4.5 | **6.1.0.11** | 4 | 99.46 | 60 | 35 ms | 99.91 | 100.46 | 4 | 48.43 |

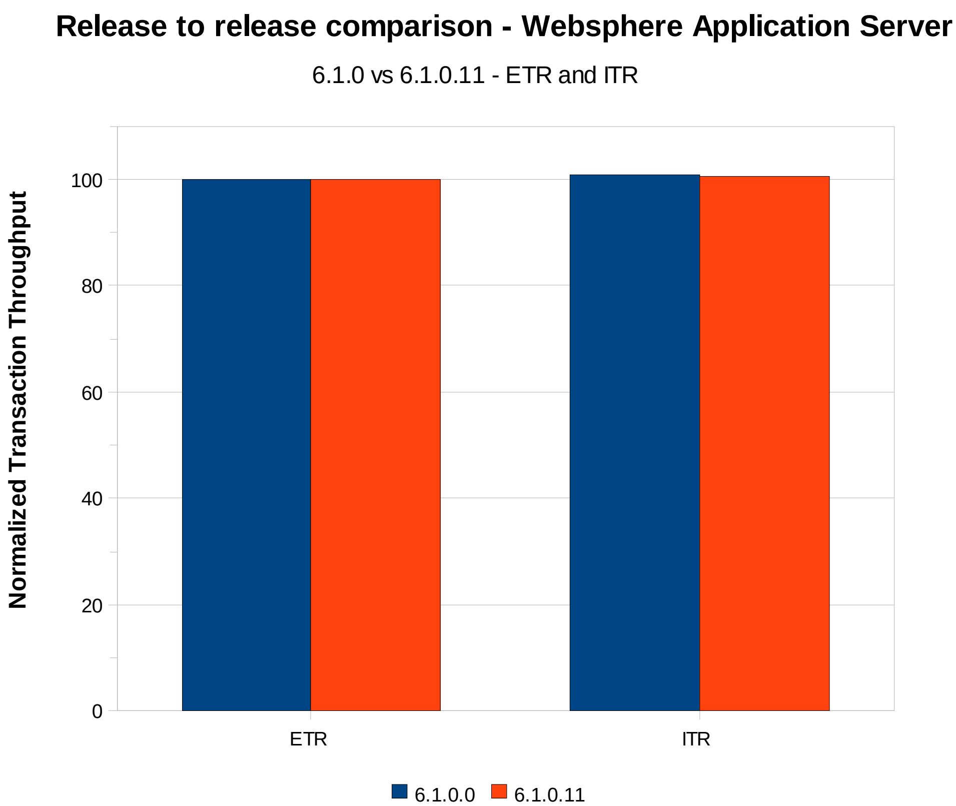


Figure 3. Release to release comparison - WebSphere Application Server

**Observations**

Both levels of code are very close in throughput characteristics. WebSphere Application Server version 6.1.0.0 consumes the same amount of WebSphere Application Server CPU resource as level 6.1.0.11. We show the ETR value for code level 6.1.0.0 as 100%, the other ETR and ITR values are shown as their relation to this value. The values for ITR, ETR, are nearly identical, with less than 1% difference between the two code levels. For this study we did not provide analysis for results with less than a 2% difference because the results might have been a run to run variance rather than a performance difference.

**Conclusion**

From a performance perspective, WebSphere Application Server version 6.1.0.11 behaves the same as version 6.1.0.0. The newer code had no impact on the performance characteristics of our workload. We always want to run with the latest code release when possible, so all further runs were performed using WebSphere Application Server version 6.1.0.11, the newer code release.

## RHEL 4.5 Trade tuning variations

The purpose of these test runs was to adjust different tuning parameters and to measure how the throughput was affected.

Our tuning variations included:

- Disk tuning variation: impact of 2105 - F20 disk latencies
- Tuning variation: Linux network settings

We applied the appropriate tuning parameters for each test. After comparing our measurements with different tuning variations, we found the optimum operating parameters for the Trade workload in our test environment and developed best practices recommendations.

### *Disk tuning variation: impact of 2105 - F20 disk latencies*

**Test case description**

These tests were performed to evaluate how the latencies from our physical disk devices affected the Trade results. To show what happens in an environment with very low disk latencies, we used a RAM disk for the database files. This test was performed to determine if our disk configuration was a performance bottleneck. Other than the RAM disk used as a disk for the database files, the Trade test setup was identical to all our other test runs. The database was on Linux on System z. For more information on our disk setup see Configuring the xpram device on RHEL

Table 4. Disk tuning variation: impact of 2105 - F20 disk latencies test runs

| | | **WebSphere Application Server** | | **Trade Results** | | | **DB2 Database** | |
|---|---|---|---|---|---|---|---|---|
| Distro | Disk | #CPU | %CPU | #Users | Resp | %ETR | #CPU | %CPU |
| RHEL 4.5 | *RAM* | 8 | 83.79 | 20 | 7 ms | 100 | 4 | 62.45 |
| RHEL 4.5 | *RAM* | 8 | 92.96 | 40 | 12 ms | 113 | 4 | 73.68 |
| RHEL 4.5 | *RAM* | 8 | 93.77 | 60 | 18 ms | 113 | 4 | 74.87 |
| RHEL 4.5 | *RAM* | 8 | 93.15 | 80 | 25 ms | 112 | 4 | 75.41 |
| RHEL 4.5 | *F20* | 8 | 91.61 | 60 | 18 ms | 111 | 4 | 77.48 |

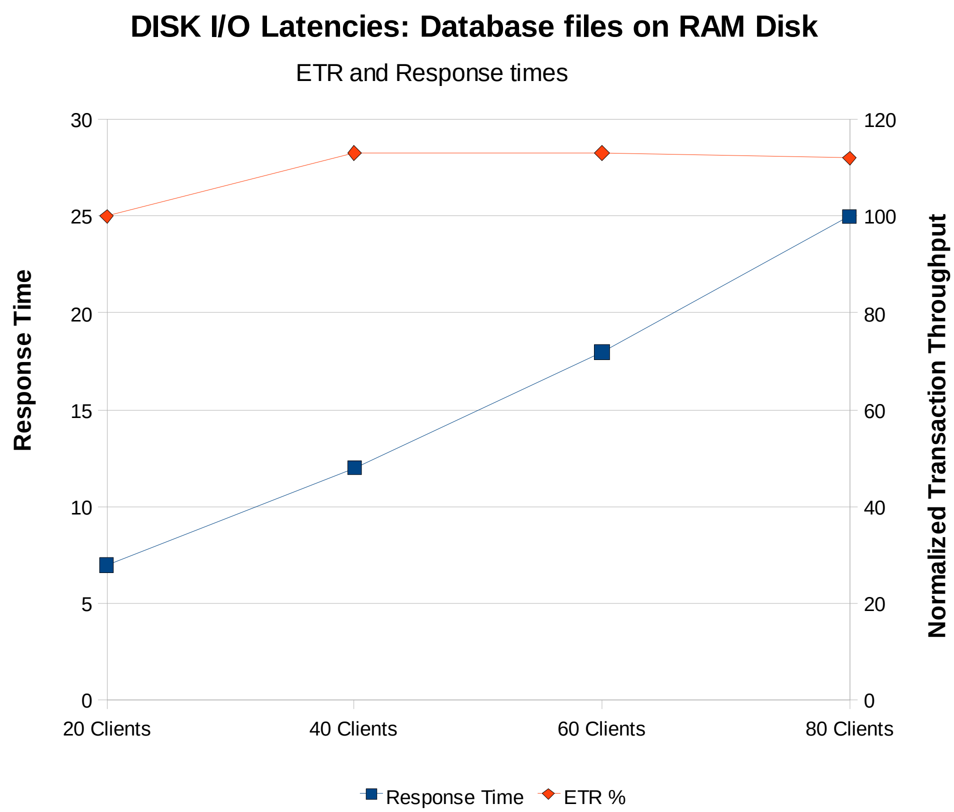


Figure 4. ETR contrast to response time as the number of users is increased

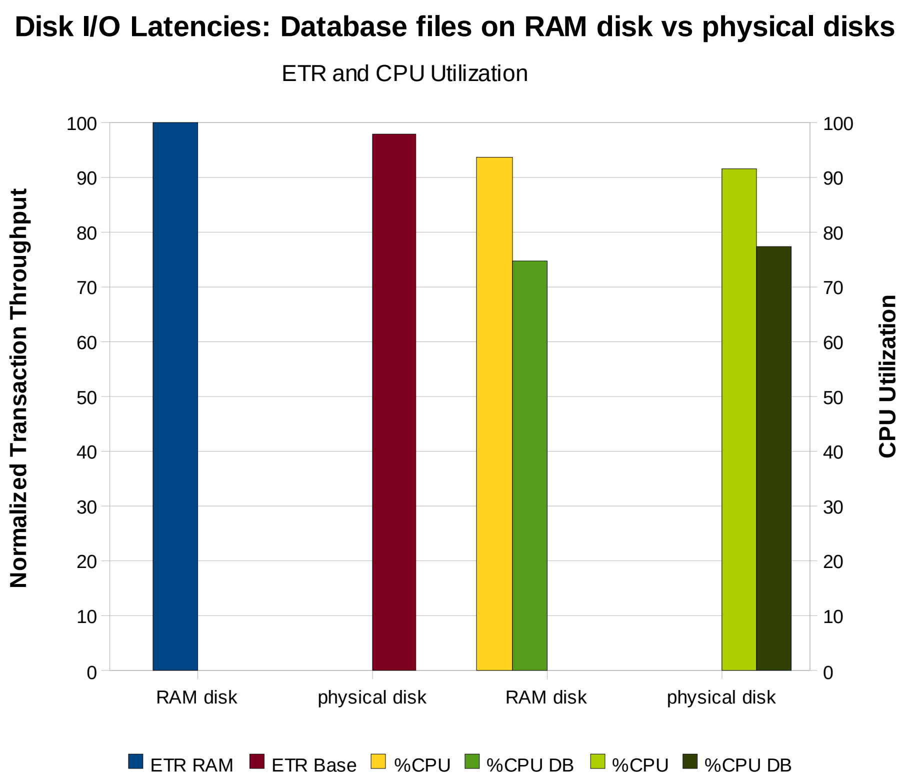


Figure 5. ETR and CPU utilization for 60 users comparing database files on RAM disk to physical disks on an F20 ESS

### Observations

Figure 4 shows throughput and latency characteristics as the number of workload generating clients (trade users) is increased to raise the load. With 80 clients the test configuration is over driven and throughput decreases. The load on the system should always be monitored so that this condition does not occur as throughput is degraded. Figure 5 shows that our results with 60 Trade users using a RAM disk for the database files is very similar to the runs with the physical F20 ESS disk run. The ETR was 2% higher using a RAM disk and the WebSphere Application Server CPU utilization was 2.4% greater. However, the test done with the RAM disk did show slightly less database CPU consumption, 3.5% less than the runs with physical disks. The response times were equal for both runs, 22 ms.

The response time increases in a linear manner when we add users. 60 users give us the most throughput, while 80 users overdrives our test setup and reduces throughput. Going from 40 to 60 users increases throughput very slightly, about 1%, but latency increases 50%.

**Conclusion**

The results with the database files on a RAM disk are similar to runs using only F20 disk devices. This shows that, in our environment, the configuration of the disk I/O subsystem is not a performance bottleneck. Verifying this fact was the only purpose of this test. Our results confirm that the change from the 2105-F20 to 2105 800 has no significant impact on our results.

Note: This was done just for testing purposes. Never use a non-persistent disk device for any component of a database!

### *Tuning variation: Linux network settings*

**Test case description**

Linux network settings were changed to evaluate the impact on throughput. We changed the buffer_count setting for all involved network connections (the private OSA device and the HiperSockets on both WebSphere Application Server and DB2). For information on how to change your buffer settings see Changing buffer settings of the network interfaces.

We ran the Trade workload in our 3 tier environment using RHEL 4.5 and also used the RHEL 4.5 database on Linux. We then compared the output of the run that used the buffer count of 128 to the base run with the default buffer count of 16.

Table 5. Tuning variation: Linux network settings test cases

| | | **WebSphere Application Server** | | **Trade Results** | | | **DB2 Database** | |
|---|---|---|---|---|---|---|---|---|
| Distro | Buffer count | #CPU | %CPU | #Users | Resp | %ETR | #CPU | %CPU |
| RHEL 4.5 | *128* | 8 | 92.54 | 60 | 18 ms | 100.0 | 4 | 78.56 |
| RHEL 4.5 | *16 (default)* | 8 | 91.61 | 60 | 18 ms | 97.5 | 4 | 77.48 |

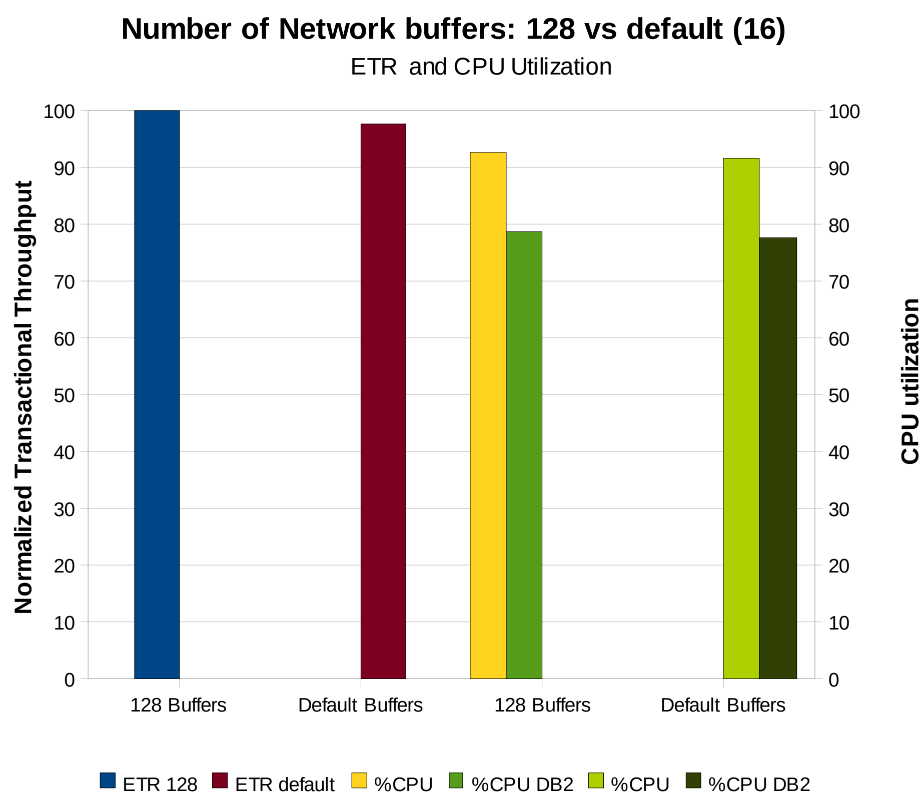


Figure 6. ETR and CPU utilization for 60 users comparing a buffer count of 128 to our default buffer count of 16

### Observations

Figure 6 shows that our results from the 60 user run, with a buffer count of 128, is very close to that of our 60 user run with a buffer count of 16. The transaction throughput with the highest number of buffers was 2.5% higher and the WebSphere Application Server percent CPU consumption was 1% greater than with the default of 16. This gave us nearly identical ITRs for the two runs. The run with 128 buffers also consumes 1.4% more CPU on the DB2 LPAR.

### Conclusion

The higher number of buffers for the network interface to the database provides a slightly higher transactional throughput. Transferring the effort for checksumming to the network interface has only an impact on the OSA card, and no impact on a HiperSockets interface. Because HiperSockets is a virtual network interface using memory to memory transfers in place of packets on a LAN interface, it does not provide the required hardware to realize checksumming offload.

## RedHat RHEL release to release comparison

The purpose of this test was to determine how the WebSphere environment performs on the various levels of RHEL. We wanted to compare RHEL 5.0 with RHEL 4.5.

For these runs we used Trade with the best tuning variations measured on our previous tests (see RHEL 4.5 Trade tuning variations).

### Test case description

These tests used the Trade workload in our 3 tier environment with both systems, the WebSphere Application Server and the database, once on RHEL 4.5 and once on RHEL 5.0. We used the tuning variations measured on our previous tests (see RHEL 4.5 Trade tuning variations):

- Increased network buffer count
- Hardware checksumming

We ran tests using RHEL 4.5 and RHEL 5.0 to the DB2 for Linux database. These runs set the baseline for future runs comparing different code levels.

*DB2 database on Linux*

Table 6. RHEL 4.5 AND RHEL 5.0 comparison

| | WebSphere Application Server | | | Trade Results | | | DB2 Database | |
|---|---|---|---|---|---|---|---|---|
| Distro | #CPU | %CPU | CPI | #Users | Resp | %ETR | CPU% | %CPU |
| RHEL 4.5 | 8 | 98.35 | 2.11 | 60 | 18 ms | 100.0 | 4 | 40.57 |
| RHEL 5.0 | 8 | 99.07 | 2.13 | 60 | 17 ms | 100.3 | 4 | 47.5 |

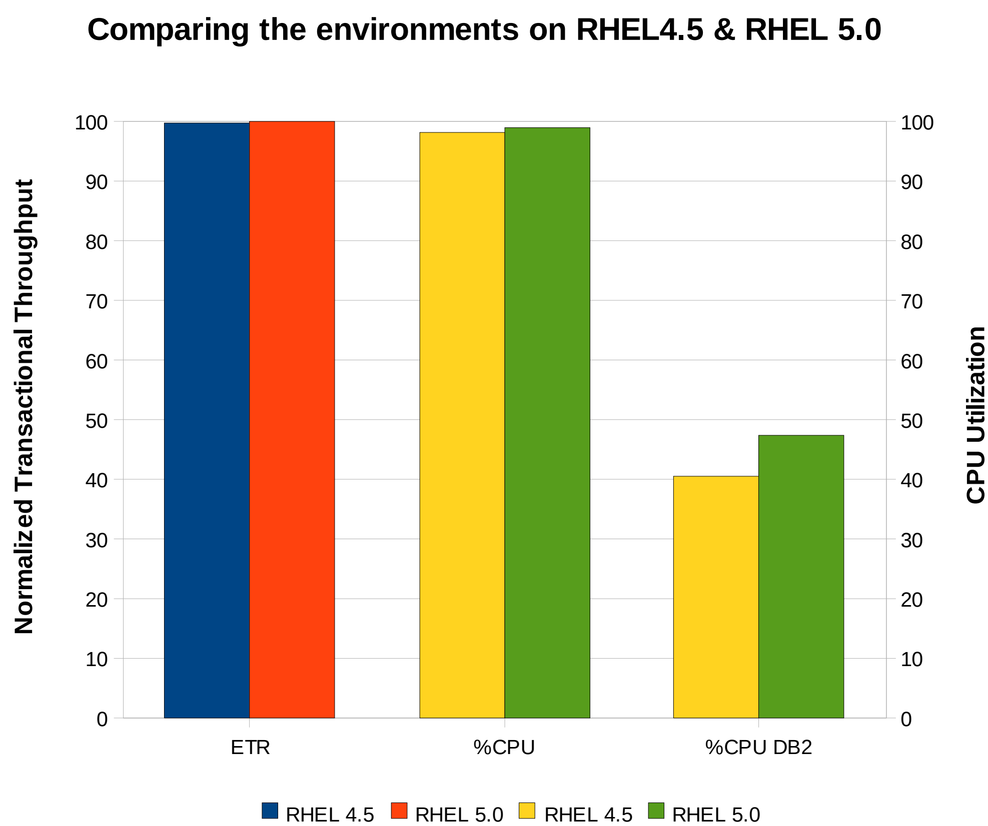


Figure 7. RHEL 4.5 and RHEL 5.0 on a Linux database runs showing ETR, ITR, and CPU consumption comparisons

**Observations**

Both the RHEL 4.5 and RHEL 5.0 environments have throughput characteristics which are very close except for the DB2 for Linux CPU utilization. This means that when WebSphere Application Server and the database are on RHEL 4.5, it requires significant less CPU on the database for approximately the same throughput as RHEL 5.0.

**Conclusion**

From a performance perspective, the performance for the environment with WebSphere Application Server on RHEL 4.5 or RHEL 5.0 is the same. However, the database requires less capacity from eight CPUs when the environment runs on RHEL 4.5. At the moment it is not clear what causes the difference.

## RHEL 5.0 scalability

The purpose of these test runs was to measure the scalability of RHEL 5.0 with one, two, four, and eight CPUs on the WebSphere Application Server LPAR. For each CPU configuration we scaled the number of clients to reach the maximum throughput. The database for each run remained constant at eight CPUs.

Note: The WebSphere Application Server CPUs and the database CPUs were in separate LPARs and were never used as a shared resource.

### Test case description

We ran the Trade workload in our 3 tier environment using the tuning parameters, we had determined, which produced our best throughput, utilizing different databases (see RHEL 4.5 Trade tuning variations). Tuning variations we used included:

- Increased network buffer count
- Hardware checksumming

We ran CPU scaling runs using RHEL 5.0 for both the WebSphere Application Server and the DB2 for Linux v9.2 LPAR. For each CPU scaling step, we scaled the number of workload generating clients to reach the maximum throughput.

Table 7. CPU scaling runs with WebSphere Application Server and database on RHEL 5.0

| | WebSphere Application Server | | Trade Results | | | | DB2 Database | |
|---|---|---|---|---|---|---|---|---|
| Distro | #CPU | %CPU | #Users | Resp | %ETR | %ITR | #CPU | %CPU |
| RHEL 5.0 | *1* | 99.96 | 5 | 11 ms | 100 | 100 | 8 | 6.52 |
| RHEL 5.0 | *2* | 99.60 | 10 | 11 ms | 199 | 200 | 8 | 13.11 |
| RHEL 5.0 | *4* | 99.30 | 20 | 11 ms | 391 | 395 | 8 | 25.58 |
| RHEL 5.0 | *8* | 99.07 | 60 | 17 ms | 750 | 757 | 8 | 47.4 |

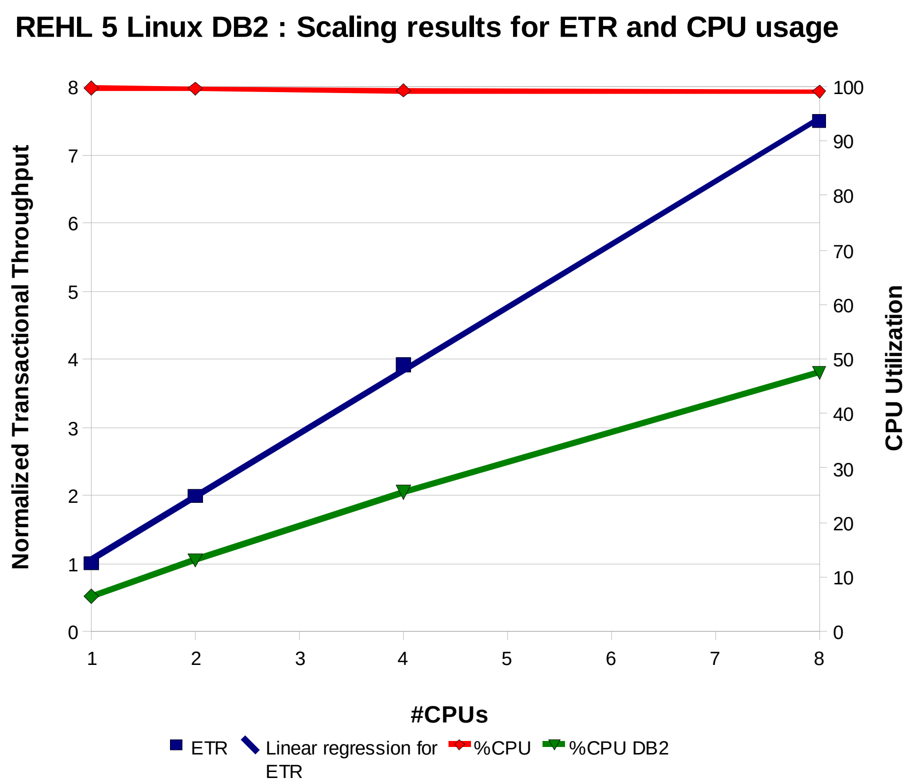


Figure 8. RHEL 5.0 ETR, WebSphere Application Server CPU percent, database CPU percent scaling results

**Observations**

Figure 8 shows the maximum throughput value (ETR) while scaling the number of CPUs. WebSphere Application Server CPU utilization and database CPU utilization were measured during our scaling runs. Our WebSphere Application Server CPU utilization was within 1% of 100% utilization for every run. The ETR slope is very linear and the scaling factors (see below) for the ETR shows this.

| | |
|---|---|
| 1 WebSphere Application Server CPU = | 1.0x |
| 2 WebSphere Application Server CPUs = | 1.99x |
| 4 WebSphere Application Server CPUs = | 3.9x |
| 8 WebSphere Application Server CPUs = | 7.5x |

The slope of the database CPU utilization is also very linear and the WebSphere Application Server is always fully utilized.

This data shows a very consistent trend for the amount of database transactions per database CPU used from one WebSphere Application Server CPU to eight WebSphere Application Server CPUs Trade runs.

**Conclusion**

These results show linear scaling for ETR and database CPU utilization for our environment on RHEL 5.0 with the DB2 for Linux database. We were able to saturate the WebSphere Application Server CPUs for each scaling point. The RHEL 5.0 system with the DB2 for Linux database was also able to handle the increase in transactions in a linear manner. This workload scaled very well on the RHEL 5.0 distribution with the hardware and software used.

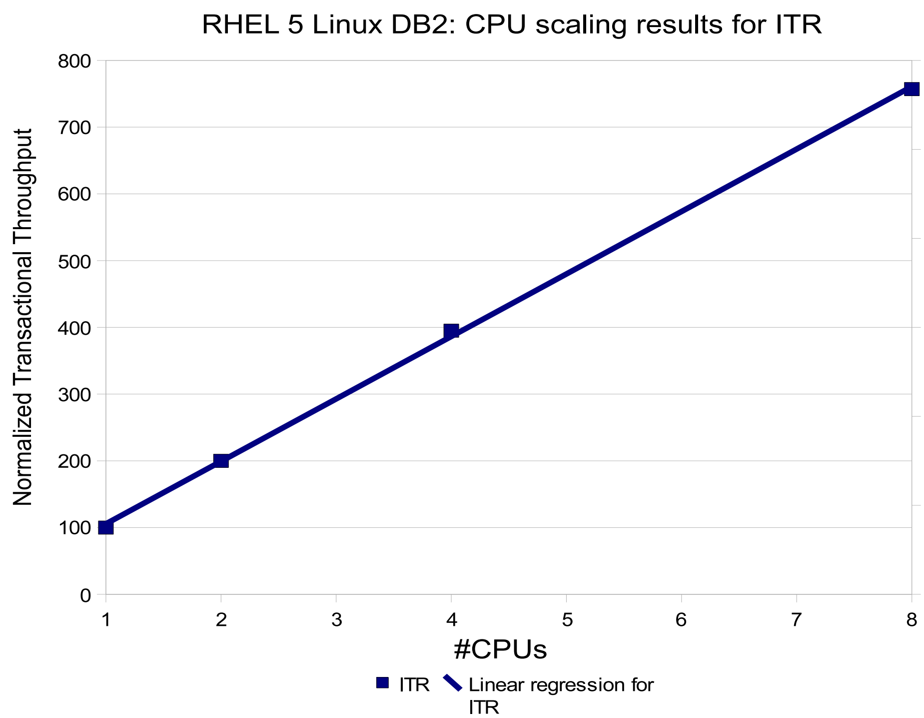


Figure 9. RHEL 5.0 ITR scaling results for Trade 6 on Linux DB2 and the regression line to depict the linear behavior

**Observations**

Figure 9 shows the ITR measured during our scaling runs. The ITR results can be shown as scaling factors to illustrate their linearity. We started with the transactions per second measured with one WebSphere Application Server CPU and gave that a weight of 1x. The remainder of the measurement points are calculated by taking the ITR and dividing it by the number of transactions measured for the 1 CPU run.

| | |
|---|---|
| 1 WebSphere Application Server CPU = | 1x |
| 2 WebSphere Application Server CPUs = | 2x |
| 4 WebSphere Application Server CPUs = | 3.95x |
| 8 WebSphere Application Server CPUs = | 7.57x |

## Conclusion

The CPU scaling results for WebSphere Application Server on RHEL 5.0 with the DB2 for Linux database show a linear scaling rate for the CPU cost of the transactions with the Trade benchmark. The ideal behavior is a scaling factor identical to the CPU scaling factor. Our eight CPU run is only 3.5% lower than a perfect scaling factor of eight. This demonstrates that the CPU power of the additional CPUs can be used to drive the workload without increasing SMP overhead.

IBM Corporation
New Orchard Rd.
Armonk, NY 10504
Printed in the United States of America, 09/2008


ZSW03041-USEN-00