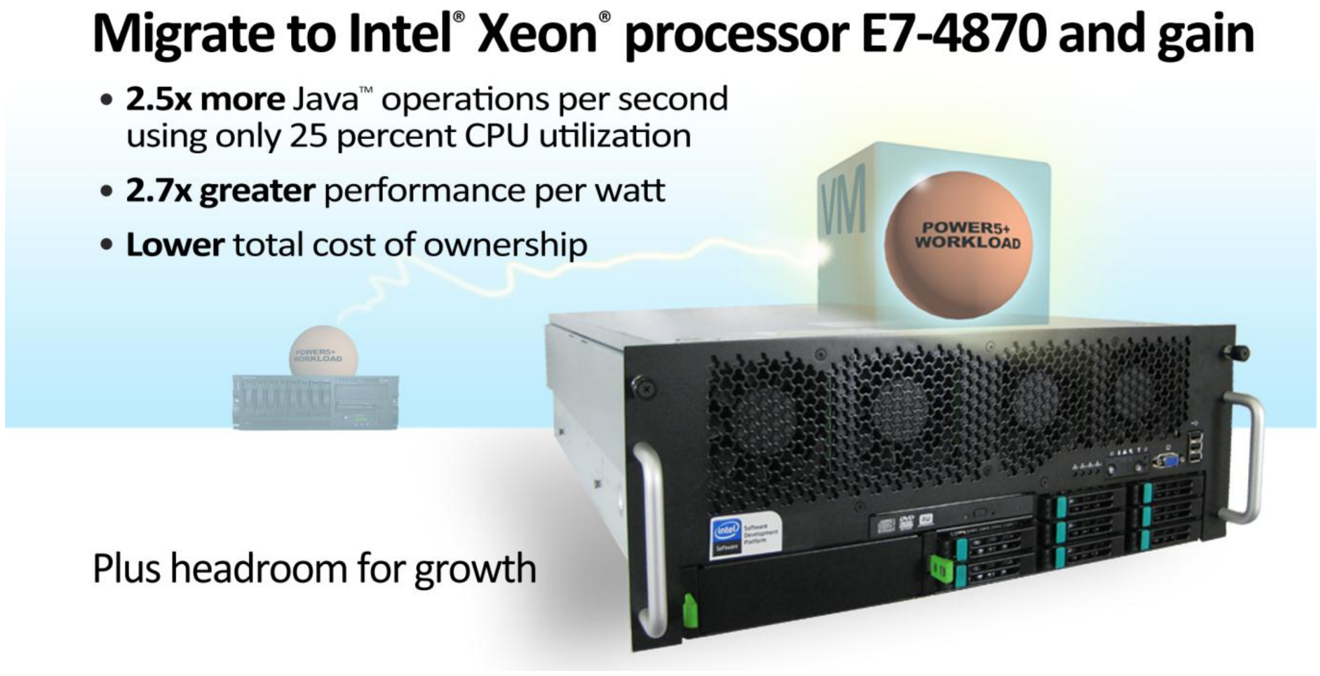
END-TO-END SOLUTION: BENEFITS OF MIGRATING FROM AN IBM POWER5+ PROCESSOR-BASED SERVER TO AN INTEL XEON PROCESSOR E7-4870-BASED SERVER



Migrate to Intel® Xeon® processor E7-4870 and gain
• 2.5x more Java™ operations per second using only 25 percent CPU utilization
• 2.7x greater performance per watt
• Lower total cost of ownership
Plus headroom for growth
VM
POWER5+ WORKLOAD

Your company requires high-performance, energy-efficient servers that can reliably run all your business-critical applications while still having headroom to spare for the future. Because your IBM® POWER5+™ processor-based servers are expensive to run and nearing their maximum performance capacity, you know that you will soon migrate your workload to new servers. Newer Intel® Xeon® processor-based servers deliver superior performance on an industry-standard platform while using far fewer system resources and less energy than IBM POWER5+ processor-based IBM System p5® 550Q servers, and have enough remaining headroom to let you consolidate multiple additional POWER5+ processor-based servers onto them.

Principled Technologies tested a four-socket Intel Xeon processor E7-4870-based server running a custom workload built on a leading Java-based infrastructure benchmark, one that simulates the flow of data through a Web-based order processing manufacturer, supply chain management, order/inventory system, and sales endpoint.

In our tests, the Intel Xeon processor E7-4870-based server yielded 2.5 times better performance and 2.7 times greater performance per watt than the IBM POWER5+ processor-based server. It did so while using only 25 percent of its processor resources (leaving 75 percent free for future growth). Upgrading to a new Intel Xeon processor E7-4870-based server costs considerably less than upgrading to a new IBM POWER7 processor-based server. Based on our research*, the Intel Xeon processor E7-4870-based server costs $31,127.10, while a comparably equipped IBM POWER7 processor-based server costs $243,392.00.

Based on our comparison, Intel Xeon processor E7-4870-based servers can provide your company with a true end-to-end solution, one that can replace your existing systems with lower operating costs found in industry-standard servers while leaving room for further expansion.

*See our Tests results in more detail section

A PRINCIPLED TECHNOLOGIES TEST REPORT
Commissioned by Intel Corp. September 2011

MIGRATE TO A STRONG SOLUTION, WITH HEADROOM TO SPARE

Power matters

As data centers become more power hungry, it's helpful to keep the following considerations in mind:

- **A recent Stanford University study[1] on datacenter electricity shows that, for every kWh used by the server, there is approximately another kWh used by the datacenter infrastructure consumed for cooling. Thus, energy-efficient servers can lower datacenter power usage.**
- **Power and cooling shortages are often the cause for server downtime; reducing server energy consumption minimizes this risk.**

Choosing the best end-to-end solution

When it comes time to migrate your current workload to new servers, you want a powerful end-to-end solution: a front-end server to handle your applications and a back-end server to support your database requests. For the end-to-end solution to function properly, both servers must perform well, even during heavy workloads, and make efficient use of power. To support your expanding business, both servers should have substantial headroom remaining, enough that they can handle even larger future workloads.

Can you find a cost-effective end-to-end server solution that achieves powerful performance and maintains generous headroom? Can you find an industry-standard Intel Xeon processor-based server to meet this need? If you choose servers powered by the new Intel Xeon processor E7-4870 and E7-8870, the answer is yes.

To support this claim, we used a Java-based benchmark to test the end-to-end performance of the Intel Xeon processor-based solution using the Intel Xeon processor E7-4870-based server as a front-end application server and the Intel Xeon processor E7-8870-based server as the back-end database server. We compared the Intel solution against an IBM POWER processor-based solution using two POWER5+-based servers, one for the application server and one for the database server. The Java workload gives operations-per-second results for the front-end application server's performance, but utilizes the back-end database server to test the total end-to-end solution, which can significantly influence the final scoring of the benchmark result.

Performance improvement

Figure 1 shows the Java operations per second and processor utilization of both application servers during testing. The Intel Xeon processor E7-4870-based server delivered 2.5 times more Java operations per second than the IBM POWER5+ processor-based server, while using 25 percent of its processor, or CPU, capacity, which demonstrates room to grow in the future. The IBM POWER5+ processor-based server used 93 percent of its processor capacity, indicating that it could not support heavier workloads as your needs grow.

[1] http://iopscience.iop.org/1748-9326/3/3/034008/

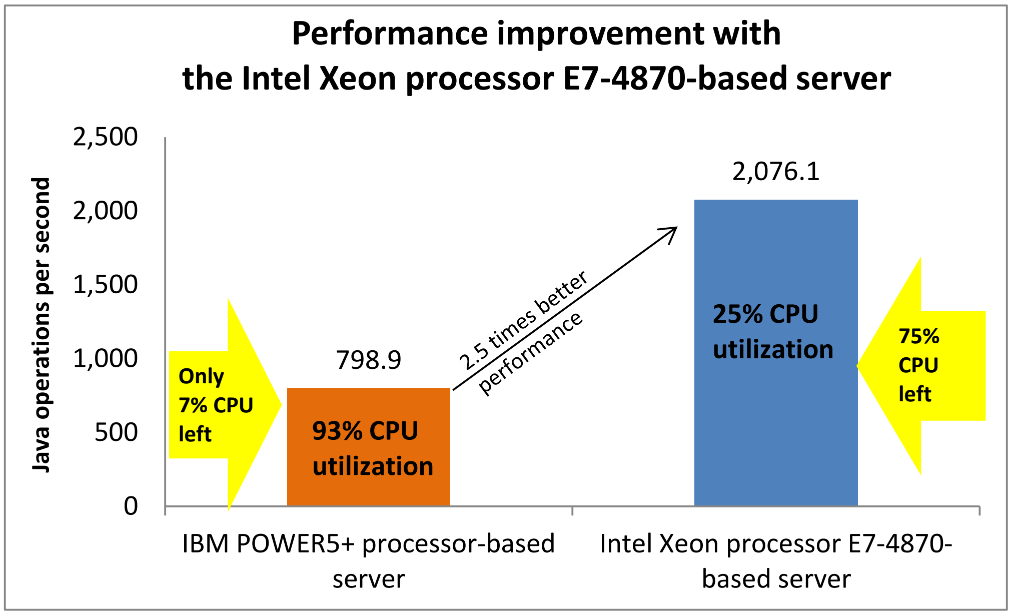


Figure 1: Performance improvement of the Intel Xeon processor E7-4870-based server over the IBM POWER5+ processor-based server. Higher performance numbers are better, while lower processor utilization numbers are better.

Performance-per-watt improvement

Figure 2 shows the performance-per-watt (PPW) and processor utilization of both application servers during testing. The Intel Xeon processor E7-4870-based server delivered 2.7 times greater performance per watt than the IBM POWER5+ processor-based server, while using 25 percent of its processor capacity. This leaves 75 percent of the processor free for future growth or to support additional VMs.

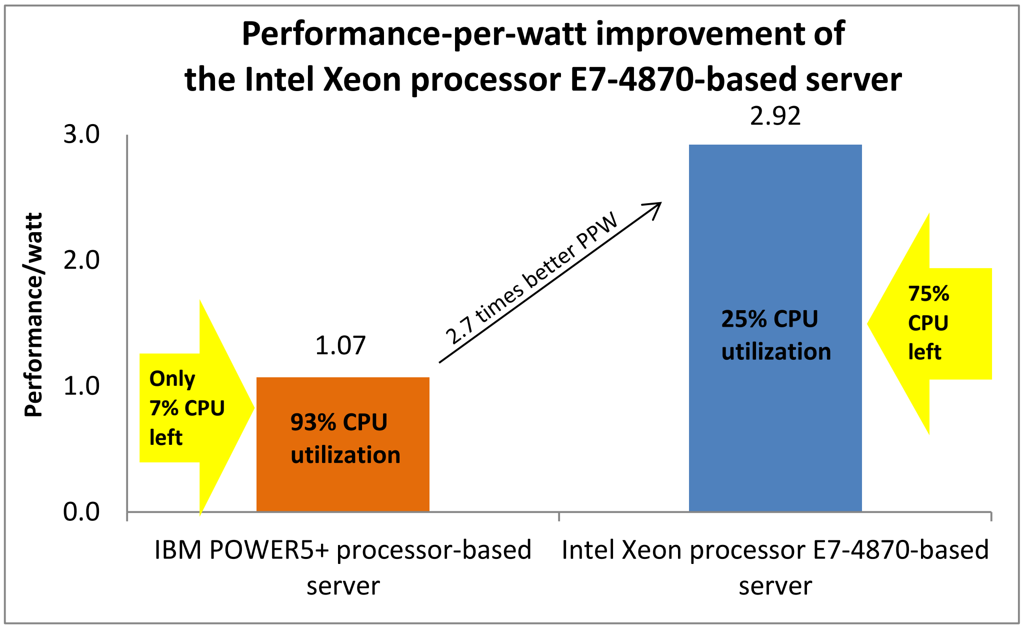


Figure 2: Performance-per-watt improvement of the Intel Xeon processor E7-4870-based server over the IBM POWER5+ processor-based server. Higher performance-per-watt numbers are better, while lower processor utilization numbers are better.

Cost savings

Cost impact is always a consideration when updating your equipment. IT has to consider whether the cost of upgrading to the same RISC architecture is more cost-effective than that of migrating to an Intel Xeon processor-based solution, one that has shown to provide near-equivalent performance in many industry-standard benchmarks. We investigated the options an IT decision maker would have when considering the cost of upgrading from an IBM POWER5+ processor-powered IBM System p5 550Q server to two different newer front-end servers, the IBM POWER7 processor-powered IBM Power 750 Express server or the Intel Xeon processor E7-4870-powered server. Note that many vendor options are available, and that we used the Quanta QCI QSSC-S4R for generic consideration.

Figure 3 compares the costs of these similarly configured servers. The IBM POWER7 processor-powered IBM Power 750 Express server costs over seven times more than the Intel Xeon processor E7-4870-powered Quanta QCI QSSC-S4R server system. While this represents substantial savings on an individual server, replacing both servers in your end-to-end solution can provide far greater savings.

Note: We do not include performance results comparing the IBM POWER7 processor-powered server and Intel Xeon E7-4870-powered server, but the servers are similarly configured. See the test results section of this report for the specifications of both servers.

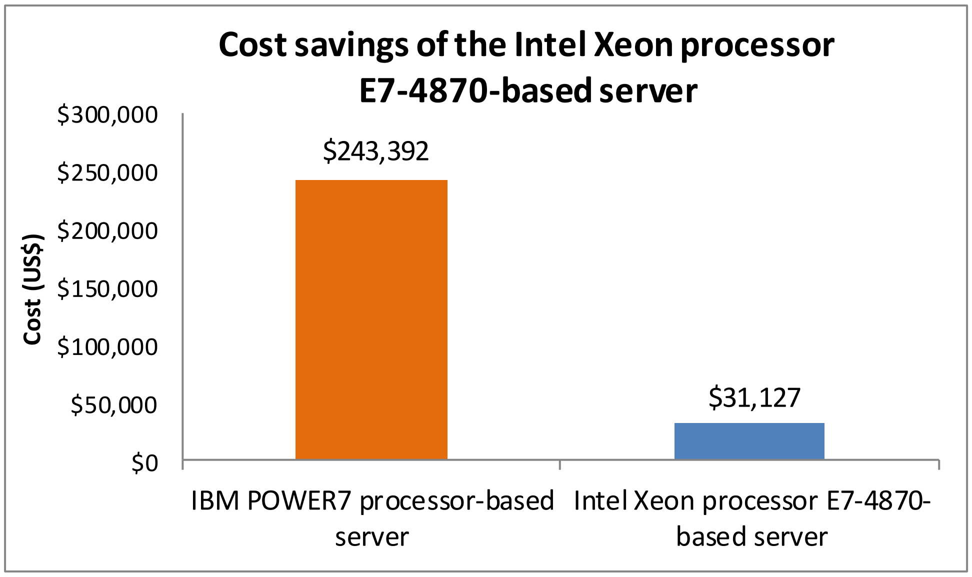


Figure 3: A new Intel Xeon processor E7-4870-based server costs a fraction of what an IBM POWER7 processor-based server costs. Lower numbers are better.

Upgrading your end-to-end solution

The results in this report show the performance-per-watt advantage of a new front-end application server. However, it is important to note that the performance of the back-end database server also matters. If the database server is running slowly, it cannot provide the necessary information promptly; this slows the application server's performance. With this in mind, the results in this report show the advantage of an Intel end-to end-solution over an existing IBM end-to-end solution.

Migration considerations

As you plan your migration from the older IBM POWER5+-based platform to the newer Intel Xeon processor E7-4870- and E7-8870-based platform, consider the following:

- **Preparing your data.** Before moving application data, back it up or export it in a portable format. Copying only the raw data files will lead to incompatibility issues. Additionally, check to see if your AIX applications have a Linux equivalent or can be recompiled for Linux, or that compatible programs exist. While both AIX and SUSE® Linux Enterprise are Unix-based operating systems, they can differ in terms of commands and configuration files. See http://bhami.com/rosetta.html for a reference guide to commands.
- **Database size/storage.** Ensure that you have sufficient storage to accommodate your data (which includes both existing data and new data that you will convert).
- **Possible downtime.** Prepare for possible downtime if you are migrating large amounts of complex data.
- **Method for migrating the database.** If you do not have a database migration tool, you will need a manual migration process to export and import data, through the use of command-line tools on each database server.

WHAT WE TESTED

Intel Xeon processor E7-4870 and E7-8870

The Intel Xeon processor E7-4870 and E7-8870 are 10-core processors with hyper-threading, for 20 total logical processors per socket. The Intel Xeon processor E7-4870 is a quad-socket server processor, while the Intel Xeon processor E7-8870 is an eight-socket server processor. These industry-standard processors provide significant consolidation headroom of business-critical applications via virtualization. They also offer a threefold performance increase on leading Java benchmarks compared to earlier processor generations. Intel research suggests that one new server powered by an Intel Xeon processor E7-4870 can replace 20 single-core servers, and that one new server powered by an Intel Xeon processor E7-8870 can replace even more single-core servers. These processors feature up to 30 MB of L3 cache and reduced power consumption via power- saving modes such as Enhanced Intel SpeedStep® Technology. They also feature performance increases via the Intel Turbo Boost feature, which enables dynamic clock speed increases when the workload requires it.

To learn more about the Intel Xeon processor E7-4870, visit http://ark.intel.com/products/53579/Intel-Xeon-Processor-E7-4870-%2830M-Cache-2_40-GHz-6_40-GTs-Intel-QPI%29.

To learn more about the Intel Xeon processor E7-8870, visit http://ark.intel.com/products/53580/Intel-Xeon-Processor-E7-8870-(30M-Cache-2_40-GHz-6_40-GTs-Intel-QPI).

IBM POWER5+ processor

The IBM POWER5+ is a quad-core processor that supports simultaneous multithreading with two threads, for eight total logical processors per socket. This allows a single quad-core POWER5+ processor to be read by the operating system as an eight-way symmetric multiprocessor.

To learn more about the IBM POWER5+ processor, visit http://www.redbooks.ibm.com/redpapers/pdfs/redp4150.pdf.

Quanta QCI QSSC-S4R server

For our application server with the Intel Xeon processor E7-4870s, we used the Quanta QCI QSSC-SW4R server. Quanta designed the QCI QSSC-S4R server for high-performance computing environments, and worked closely with Intel on the development of this server. The QCI QSSC-S4R provides the power suitable for large enterprise datacenters in the form of a rack-mounted server that features up to 1 TB of system memory, 10 PCIe expansion slots, up to 4 TB of internal storage with a maximum of eight 2.5-inch hot-swappable SATA/SAS hard drives, and support for four processors (40 independent processor cores). The QCI QSSC-S4R also features redundant cooling and power supply for increased reliability.

To learn more about the Quanta QCI QSSC-S4R server, visit http://www.qsscit.com/en/01_product/02_detail.php?mid=27&sid=125&id=126&qs=50

NEC Express5800/A1080a server

We used the NEC Express5800/A1080a as our back-end database server, which features the Intel Xeon processor E7-8870. The NEC Express5800/A1080a server is a fifth-generation x86 Enterprise server. It supports a maximum of 2 TB of memory and eight processor sockets. It features the Intel Xeon processor E7-8800 family of processors and Intel QuickPath Interconnect technology, which can deliver significant database performance increases compared to the previous generation. The NEC Express5800/A1080a also offers power-efficient performance due to green power and cooling technology and energy-efficient cooling supplies.

To learn more about the NEC Express5800/A1080a 8-way server, visit http://www.necam.com/servers/enterprise/A1080a.cfm.

IBM System p5 550Q server

For our IBM POWER5+ processor-based system, we used the IBM System p5 550Q server. The IBM System p5 550Q server is a scalable, multi-application server designed to offer performance, reliability, and affordability. It offers 64-bit scalability, with four- and eight-core configurations. The System p5 550Q supports up to 64 GB of memory and up to 2,400 GB of internal disk storage. It offers a browser-based Integrated Virtualization Manager to assist with the virtualization process.

To learn more about the System p5 550Q server, visit http://www-03.ibm.com/systems/power/hardware/systemp/entry/550/index.html.

IBM WebSphere

We used IBM WebSphere® as our software application server on the front-end servers we tested. IBM WebSphere refers to a range of enterprise software products that allows users to create applications and integrate them with other applications (such software is referred to as “middleware”). For testing, we used WebSphere Application Server 8.0.

To learn more about the WebSphere, visit http://www-01.ibm.com/software/websphere/#.

IBM DB2

For the back-end servers, we used IBM DB2 for the database. IBM DB2 is database server software that runs on several operating systems, including Unix, Linux®, and Windows®. DB2 is available in three editions: Express, Workgroup Server, and Enterprise Server. We used the DB2 Enterprise Server 9.7 edition.

To learn more about the BD2, visit http://www-01.ibm.com/software/data/db2/.

OUR WORKLOAD

Java benchmark

We used a workload based on a leading Java infrastructure benchmark. This benchmark measures the system performance of Java Web application and database servers. It also measures performance of storage, operating systems, software, and the system network. Specifically, the workload simulates the flow of data in an automotive dealership, the manufacturer, the supply chain, and the order/inventory system. The performance metric is operations per second.

TEST RESULTS IN MORE DETAIL

Performance results

Figure 4 presents the performance results from our benchmark, measured in operations per second. The Intel Xeon processor E7-4870-based server provided 2.5 times the performance of the IBM POWER5+-based server running the same workload.

It is important to note the processor utilization on the two application servers. In addition to providing better performance, the Intel Xeon processor E7-4870-based server used 25 percent of the system's total processor capacity. The IBM POWER5+ processor-based server used 93 percent of its total available processing power, leaving very little remaining headroom to handle additional work. Thus, the Intel Xeon processor E7-4870-based server provides superior performance with room to spare.

Figure 4 also shows the processor utilization for the back-end database servers. The Intel Xeon processor E8870-based server only used 4 percent of its total available processor utilization, while the IBM POWER5+ processor-based server used 28 percent. Even though the back-end IBM POWER5+ processor-based server does have additional processor utilization room, the application server is using most of its processor and therefore cannot handle any more work, so the IBM POWER5+ processor-based server solution is maxed out as an end-to-end solution. In contrast, the Intel Xeon processor-based solution provides better end-to-end performance, with resources to spare on both servers.

Performance results			
	Operations per second	**Performance per watt**	**Percentage processor utilization**
Intel Xeon processor E7-4870-based server (application server)	2,076.1	2.92	25
IBM POWER5+ processor-based server (application server)	798.9	1.07	93
Intel Xeon processor E7-8870-based server (database server)	N/A	N/A	4
IBM POWER5+ processor-based server (database server)	N/A	N/A	28

Figure 4: Detailed performance results for the Intel and IBM processor-based servers. Higher performance numbers are better, while lower processor utilization numbers are better.

Power results

Figure 5 presents the power results, with lower numbers being better. While the peak performance power numbers seem similar for each application server, it is important to remember that the performance increase of the Intel Xeon processor E7-4870-based application server is substantial: it provides 2.5

times the performance while using slightly less power, making its performance-per-watt advantage substantial. In addition, because the processor utilization is 25 percent of the total available processing power, it has plenty of room for additional workloads.

Figure 5 also shows the power utilization results for both database servers. While the power for the Intel Xeon processor E7-8870-based server is greater than that for the IBM POWER5+ processor-based server, it is important to note that the Intel server is an 8-socket server, while the IBM server has a dual-processor card with four cores per card. Thus, the power usage for the Intel server will be greater, but the available performance ability will also be substantially greater.

Power usage would increase with additional work on the Intel Xeon processor E7-4870-based server. However, because the fans and additional components of the server are already running, the power increase would be modest. As you increase the amount of processor utilization from additional workloads, the performance-per-watt advantages should increase as well.

Note that the idle power usage of the Intel solution can be improved further through Intel Node Manager implementation, saving potentially hundreds of thousands of kWh per year. See http://www.intel.com/content/www/us/en/data-center/data-center-management/intelligent-power-node-manager-general.html for more information.

Power results		
	Idle power (Watts)	**Peak performance power (Watts)**
Application server		
Intel Xeon processor E7-4870-based server	608	710
IBM POWER5+ processor-based server	724	748
Database server		
Intel Xeon processor E7-8870-based server	1,080	1,137
IBM POWER5+ processor-based server	747	742

Figure 5: Detailed power results for the Intel and IBM processor-based servers. Lower numbers are better.

Cost findings

Figure 6 presents the itemized cost for the Intel Xeon processor E7-4870-based server and the IBM POWER7 processor-based server. For the Intel Xeon processor E7-4870-based server, we priced a Quanta QCI QSSC-S4R server as we used for testing. The IBM Power 750 Express server we priced has specifications as close as possible to those of the Intel Xeon processor E7-4870-based server.

As the total cost shows, upgrading to the Intel Xeon processor E7-4870-based server provides substantial savings.

Note that a large amount of the IBM Power 750 Express server cost is for processor activation. This cost is necessary to use all the server's processor cores. The IBM Power 750 Express server price would be lower if an end user did not activate all cores. However, we show the price to use all available server resources because all of the Intel Xeon processor E7-4870-based server's processors are likewise available; no additional activation fee is required.

We received our pricing for the IBM Power 750 Express server in an email from authorized IBM reseller Abacus Solutions, dated November 9, 2011. We received our pricing for components in the Quanta QCI QSSC-S4R server from several sources. For the Intel Xeon processor E7-4870, we received an online quote from ServerSupply, dated November 9, 2011. For the memory and the hard disks, we received an online quote from Memory4Less, dated November 9, 2011. For the base server, storage controller, and memory risers, we received an email quote from CTL, dated November 11, 2011. We report list prices for each component.

Total cost				
	Quanta QCI QSSC-S4R server		**IBM Power 750 Express server**	
	Description	**Price**	**Description**	**Price**
Base server	Quanta QCI QSSC-S4R	$3,699.00	Model 8233-E8B	$7,532.00
Processor (Qty 4)	10-core 2.4 GHz Intel Xeon processor E7-4870	$21,360.00	8-core 3.6GHz POWER7	$70,800.00
DVD Drive	SATA	Included	SATA	$299.00
Storage controller	SAS controller	$529.00	SAS controller	$799.00
NIC	4-port 1Gb	Included	4-port 1Gb	$528.00
Power Supply	850 watt (Qty 4)	Included	1,725 watt (Qty 2)	$1,398.00
Memory riser	4 additional memory risers	$716.00	Not required	N/A
Memory	32 x 4GB DDR3 1,333 MHz	$4,204.80	32 x 4GB DDR3 1,066 MHz	$17,040.00
Hard disk (Qty 2)	146GB, 10K RPM, SAS	$618.30	146 GB, 15K RPM, SAS	$996.00
	Quanta QCI QSSC-S4R server		**IBM Power 750 Express server**	
	Description	**Price**	**Description**	**Price**
Processor activation	N/A	N/A	Processor activation for all cores	$144,000.00
Total cost		**$31,127.10**		**$243,392.00**

Figure 6: Total costs for the Intel and IBM processor-based servers. Lower numbers are better.

We explain our system specs in Appendix A and our test methodology in Appendix B.

FINAL THOUGHTS

Based on these results, industry-standard Intel Xeon processor E7-4870- and E7-8870-based servers provide businesses with a true end-to-end solution that can replace your existing systems with lower operating costs while leaving room for further expansion.

In our application server comparison, the Intel Xeon processor E7-4870-based server offered 2.7 times the performance per watt and 2.5 times better performance compared to the IBM POWER5+ processor-based server, all while only using 25 percent of its processor capacity. The Intel Xeon processor E7-4870-based server consolidates your previous system while leaving 75 percent of the processor capacity free for handling additional workloads. The idle power is lower on the Intel solution than on the IBM POWER solutions.

While our results show the advantage of upgrading to a new Intel Xeon processor E7-4870-based server for the application server, keep in mind that the back-end database server affects performance as well. Using the Intel Xeon processor E7-8870 based server as your back-end database server gives you a complete package: better performance, better performance per watt, and greater remaining headroom. Furthermore, upgrading your end-to-end solution with new Intel processor-based servers costs significantly less than upgrading to new IBM POWER7-based servers.

APPENDIX A – SERVER AND STORAGE CONFIGURATION INFORMATION

Figure 7 provides detailed configuration information about the test servers. We used two IBM POWER5+ processor-based servers, one as an application server and one as a database server, each with the same configuration. The Intel Xeon processor E7-4870-based server acted as our front-end application server, and the Intel Xeon processor E7-8870-based server acted as our back-end database server.

Servers	IBM POWER5+ processor-based server	Intel Xeon processor E7-4870-based server	Intel Xeon processor E7-8870-based server
Power supplies			
Total number	2	4	2
Vendor and model number	Delta Energy AWF-9DC-1050W	Delta DPS-850FB	NEC RH1685
Wattage of each (W)	1,050	850	2,200
Cooling fans			
Total number	4	8	16
Vendor and model number	San Ace 120 9G1212E403	Delta PFB0812DHE	Delta PFC0912DE
Dimensions (h x w) of each	4.75” x 4.75”	3.50" x 3.50"	4.0” x 4.0”
Volts	12	12	12
Amps	0.58	3.3	3.72
General processor setup			
Number of processor packages	2	4	8
Number of cores per processor package	4	10	10
Number of hardware threads per core	2	2	2
Processor			
Vendor	IBM	Intel	Intel
Model	POWER5+	Xeon E7-4870	Xeon E7-8870
Socket type	POWER5+	LGA1567	LGA1567
Core frequency (GHz)	1.65	2.40	2.40
Bus frequency	1,066 MHz	6.4 GT/s	6.4 GT/s
L1 cache (KB)	32 KB Data + 64 KB Instruction	32 KB + 32 KB (per core)	32 KB + 32 KB (per core)
L2 cache	1.9 MB (shared)	256 KB (per core)	256 KB (per core)
L3 cache (MB)	36	30	30
Platform			
Vendor and model number	IBM System p5 550Q	Quanta QCI QSSC-S4R	NEC Express5800/A1080a
Motherboard chipset	Power5	Intel 7500	Intel 7500
BIOS name and version	SF240_382 (Firmware version)	Intel QSSC-S4RQCI.01.00.S012 (03/14/2011)	NEC 02.52 R3638 (07/22/2011)
BIOS settings	N/A	Intel Virtualization Technology Enabled	Default

Servers	IBM POWER5+ processor-based server	Intel Xeon processor E7-4870-based server	Intel Xeon processor E7-8870-based server
Memory modules			
Total RAM in system (GB)	24	128	512
Vendor and model number	Samsung M338T5750CZ3-CD5M2	Hynix HMT151R7TFR4A-H9	Samsung M393B1K70CH0-YH9
Type	PC2-4200	PC3L-10600	PC3L-10600
Speed (MHz)	533	1,333	1,333
Speed in the system currently running @ (MHz)	533	1,333	1,333
Timing/Latency (tCL-tRCD-iRP-tRASmin)	4-4-4-12	9-9-9-36	9-9-9-36
Size (GB)	2	4	8
Number of RAM modules	12	32	64
Rank	Dual	Dual	Dual
Hard disk			
Vendor and model number	Seagate ST3146707LC	Seagate ST9146802SS	Seagate ST9300653SS
Number of disks in system	2	2	2
Size (GB)	146	146	300
Buffer size (MB)	8	16	64
RPM	10,000	10,000	15,000
Type	Ultra320	SAS	SAS
Storage controller			
Vendor and model number	Dual Channel Ultra320 SCSI adapter	Intel RS2BL080	LSI MegaSAS 9260-8i
Type	Integrated	PCI Express	Integrated
Firmware	070A0011	12.0.1-0045	12.0.1-0099
RAID configuration	Individual disk	RAID 1	RAID 1
Operating system			
Name	IBM AIX 7.1	VMware ESXi 5.0.0	SUSE Linux Enterprise Server 11
Build number	7100-00-01-1037	469512	2.6.32.12-0.7
File system	Jfs2	VMFS	EXT3
Language	English	English	English
Network card/subsystem			
Vendor and model number	Dual-port 1000 Base-TX adapter	Dual Port Intel 82576NS Gigabit Ethernet Controller	Dual Port Intel 82576 Gigabit Ethernet Controller
Type	Integrated	Integrated	Integrated

Servers	IBM POWER5+ processor-based server	Intel Xeon processor E7-4870-based server	Intel Xeon processor E7-8870-based server
Fibre card/subsystem			
Vendor and model number	EMULEX LP11002 (database server only)	N/A	EMULEX LPe12002
Type	PCI-X	N/A	PCI Express

Figure 7: Detailed configuration information for the test servers.

APPENDIX B – DETAILED TEST METHODOLOGY

Here, we explain the steps we followed to set up our test scenario.

For testing, we used the same benchmark workload on both systems, but increased the number of users on the Intel Xeon processor E7-4870-based server, as it could handle the additional performance. Both servers ran two instances of IBM WebSphere Application server. On the IBM POWER5+ processor-based server, we ran these two instances in a non-virtualized environment with two active network connections on the server and all 16 logical processors available for server operations, so the server used all available resources. On the Intel Xeon processor E7-4870-based server, we ran two VMs with one IBM WebSphere Application server instance per VM. Each VM had eight logical processors, so the total number of processors used would be equal the IBM POWER5+ processor-based server. We configured each VM on the Intel Xeon processor E7-4870-based server with two network connections. The Intel Xeon processor E7-4870-based server had twice the total number of network connections than the IBM POWER5+-based server, but since the IBM POWER5+-based server's processor was saturated, adding additional network adapters would not improve its performance. The Intel Xeon processor E7-8870-based database server and IBM POWER5+-based database server ran as non-virtualized servers.

Figures 8 and 9 show the layout of the test beds we used for testing.

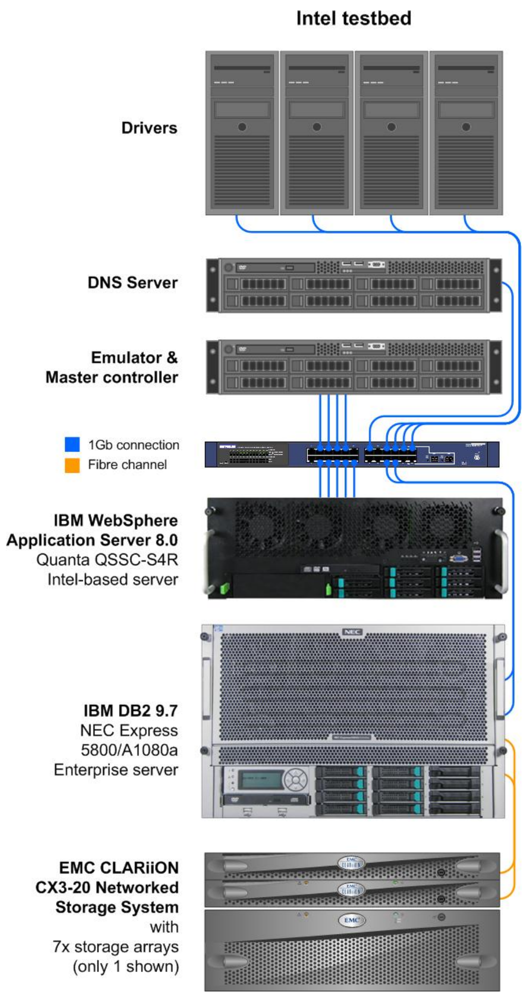


Figure 8: Intel server test bed.

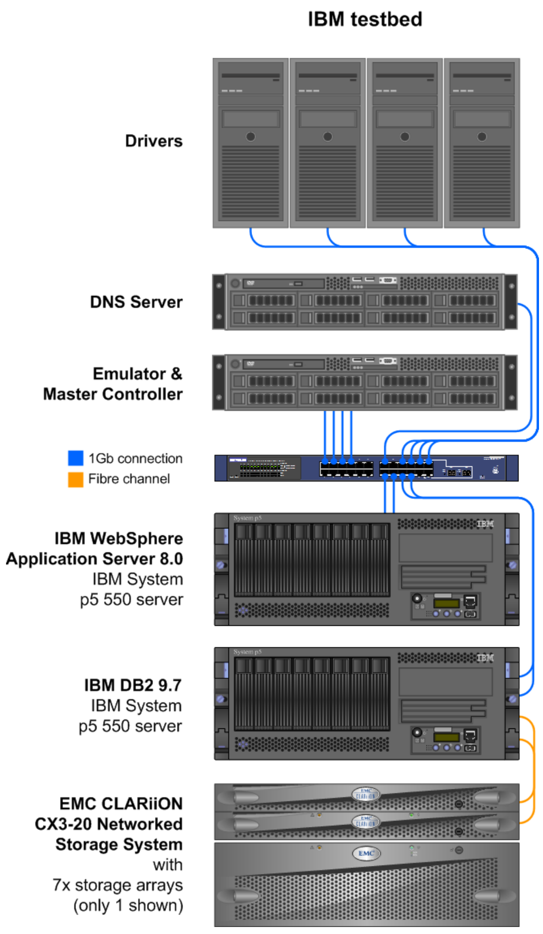

IBM testbed
Drivers
DNS Server
Emulator & Master Controller
🟦 1Gb connection
🟧 Fibre channel
IBM WebSphere Application Server 8.0 IBM System p5 550 server
IBM DB2 9.7 IBM System p5 550 server
EMC CLARiiON CX3-20 Networked Storage System with 7x storage arrays (only 1 shown)

Figure 9: IBM server test bed.

Setting up the test bed

In the steps below, we give detailed instructions of how we configured the various components of our test bed.

IP scheme used for testing

Figures 10 through 14 show the IP address scheme we used for testing.

Drivers	IP	Domain name
DRIVER 01 – NIC1	192.168.1.61	driver01-1.test.lan
	192.168.2.61	driver01-2.test.lan
	192.168.3.61	driver01-3.test.lan
	192.168.4.61	driver01-4.test.lan
DRIVER 02 – NIC1	192.168.1.62	driver02-1.test.lan
	192.168.2.62	driver02-2.test.lan
	192.168.3.62	driver02-3.test.lan
	192.168.4.62	driver02-4.test.lan

Drivers	**IP**	**Domain name**
DRIVER 03 – NIC1	192.168.1.63	driver03-1.test.lan
	192.168.2.63	driver03-2.test.lan
	192.168.3.63	driver03-3.test.lan
	192.168.4.63	driver03-4.test.lan
DRIVER 04 – NIC1	192.168.1.64	driver04-1.test.lan
	192.168.2.64	driver04-2.test.lan
	192.168.3.64	driver04-3.test.lan
	192.168.4.64	driver04-4.test.lan

Figure 10: IP and domain name information for the drivers.

Emulator	**IP**	**Domain name**
EMULATOR - NIC1	192.168.1.81	emulator01.test.lan
EMULATOR - NIC2	192.168.2.82	emulator02.test.lan
EMULATOR - NIC3	192.168.3.83	emulator03.test.lan
EMULATOR - NIC4	192.168.4.84	emulator04.test.lan

Figure 11: IP and domain name information for the emulator

Application servers	**IP**	**Domain name**
Intel AppServer server (HyperVisor)	**192.168.1.12**	**appvmhost.test.lan**
Intel AppServer (deployment manager VM)	192.168.1.120	inteldmgr01.test.lan
	192.168.2.120	inteldmgr03.test.lan
	192.168.3.120	inteldmgr02.test.lan
	192.168.4.120	inteldmgr04.test.lan
Intel AppServer VM1 - NIC1	192.168.1.121	intelapp01.test.lan
Intel AppServer VM2 - NIC2	192.168.3.123	intelapp03.test.lan
Intel AppServer VM 3- NIC3	192.168.2.122	intelapp02.test.lan
Intel AppServer VM4 - NIC4	192.168.4.124	intelapp04.test.lan
IBM AppServer server		**powerapp.test.lan**
IBM AppServer - NIC1	192.168.1.221	powerapp01.test.lan
IBM AppServer - NIC2	192.168.1.222	powerapp02.test.lan

Figure 12: IP and domain name information for the application servers.

Database servers	**IP**	**Domain name**
Intel Database server		**inteldb.test.lan**
Intel Database server - NIC1	192.168.1.111	inteldb01.test.lan
Intel Database server - NIC2	192.168.1.112	inteldb02.test.lan
IBM Database server		**powerdb.test.lan**
IBM Database server - NIC1	192.168.1.211	powerdb01.test.lan
IBM Database server - NIC2	192.168.1.212	powerdb02.test.lan

Figure 13: IP and domain name information for the database servers.

DNS Server	**IP**	**Domain name**
DNS Load Balancer	**192.168.1.5**	**dnshost.test.lan**
DNSHOST - NIC1	192.168.1.5	dnshost01.test.lan
	192.168.2.5	dnshost02.test.lan
	192.168.3.5	dnshost03.test.lan
	192.168.4.5	dnshost04.test.lan

Figure 14: IP and domain name information for the DNS server.

Setting up and configuring the EMC storage

We used an EMC® CLARiiON® Fibre Channel connected SAN for this testing. The CX3-40 has two storage processors SP-A and SP-B. We used a dual-port host bus adapters (HBA) in our database servers for testing. We cabled both HBA ports to each SP (A and B) on the SAN to balance the load between SPs. We used seven enclosures of disks on the SAN.

We created six RAID groups for our test server. Each of the six RAID groups was composed of 16 disks, and set up as a RAID 10. We then created two LUNs per RAID group. One for IBM AIX configuration and one for Intel configuration. We used two LUNs for logs, for a total of 32 disks and four LUNs for data, for a total of 64 disks on both configurations.

Setting up a controller

The IBM POWER5+ processor-based servers under test did not have a video card installed, so we performed all configuration and setup functions via telnet from a remote controller. We used a SuperMicro® SuperServer® 6024H-T as the controller and installed Windows Server 2003 R2 on it.

Installing Windows Server 2003 R2 Enterprise Edition

We used the following steps to set up Windows Server 2003 R2 on the controller.

1. Boot the server, and insert the Windows Server 2003 R2 Service Pack 2 installation DVD in the DVD-ROM drive.
2. At the Welcome to Setup screen, press Enter.
3. At the Windows Licensing Agreement screen, press F8 to agree to the terms.
4. Select the appropriate partitioned space, and press C to create a partition. (We chose to use the entire disk.)
5. Select Format the partition using the NTFS file system, and press Enter.
6. At the Regional and Language Options screen, click Next.
7. Enter Name and Organization, and click Next.
8. At the Your Product Key screen, enter your product key, and click Next.
9. At the Licensing Modes screen, click Next.
10. At the Computer Name and Administrator Password screen, choose an appropriate name and password, and click Next.
11. At the Date and Time Settings screen, change the Date & Time and Time Zone if appropriate, and click Next.
12. After the system restarts automatically, follow the prompt to log into the server.
13. At the Windows Setup screen, insert the second installation CD when prompted, and click OK.
14. At the Windows Server Post-Setup Security Updates screen, click Finish.
15. When prompted, click Yes to close the page.
16. At the Windows Server 2003 R2 Setup Wizard Welcome screen, click Next.
17. On the Windows End User License Agreement screen, select I Accept, and click Next.
18. At the Windows Server 2003 R2 Setup Summary screen, click Next.
19. At the Completing Windows Server 2003 R2 Setup, click Finish.
20. Reboot the server.

21. Log into the server after it reboots.
22. Navigate to Start→Control Panel→Network Connections, and select your network adapter.
23. Click Properties.
24. Select Internet Protocol (TCP/IP), and click Properties.
25. Enter a static IP address, subnet mask, default gateway, and DNS address. Click OK.
26. On the Network Properties window, click OK.
27. On the Network Status window, click Close.
28. Navigate to Start→All Programs→Windows Update on the Security Warning screen, and click Install.
29. On the Internet Explorer warning about IEESC being enabled, check Do not show this warning, and click Ok.
30. At the Windows Update screen, click Install Now.
31. At the Windows update screen, click Express.
32. At the Internet Explorer prompt, check In the future do not show this message, and click Yes.
33. At the Express results screen, click Install Updates.
34. At the Installing updates screen, read the license terms, and click I Accept.
35. At the Install Windows Internet Explorer 8, click I do not wish to participate, and click Next.
36. At the license terms screen, click I accept.
37. At the latest updates screen, click Next.
38. At the Installation complete screen, click Restart Now.
39. At the Log On screen, enter username and password, and click OK.
40. Navigate to Start→All Programs→Windows Update.
41. At the Setup Windows IE 8, click Ask me later.
42. At the Keep your computer up to date, click Express.
43. At the Express results, click Install Updates. (We installed all available critical updates.)
44. At the installation complete screen, click Restart Now.
45. At the Log On Screen, enter username and password, and click OK.

Installing Cygwin

We used Cygwin to connect to the IBM servers via telnet. We installed the latest version of Cygwin from http://www.cygwin.com/ on the controller and used it to communicate with the IBM POWER5+ processor-based servers.

1. Download setup.exe from http://www.cygwin.com/setup.exe to your system.
2. Navigate to the setup.exe, and double click it to execute.
3. At the Welcome screen, click Next.
4. At the Choose A Download Source page, select Install from Internet, and click Next.
5. Select a path for the root directory at the Choose Installation Directory page, and click Next. (We kept the default location.)
6. At the Select Local Package Directory, keep the default, and click Next.
7. Select your Internet connection at the Select Connection Type screen, and click Next. (We chose Direct Connection.)
8. Select a download site at the Choose Download Site, and click Next.
9. At the Select Packages screen, keep all defaults, but select all X11 and Network packages, and click Next. The packages will download and install automatically.
10. At the Installation Status and Create Icons screen, keep the defaults, and click Finish.

Setting up the IBM POWER5+ processor-based servers

We configured both servers with a fresh installation of AIX 7.1. We used the following steps to configure the servers for testing.

Configuring the IBM POWER5+ processor-based server network connections

Before connecting to the IBM POWER5+ processor-based server via Telnet, we had to configure the network ports. To configure the network ports, we connected to server via serial connection and ran PuTTY. We downloaded the latest version of PuTTY from http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html. We used the following steps to configure the network adapters on both servers.

1. Double-click the PuTTY executable.
2. On the PuTTY configuration screen, select serial under connection type.
3. Enter `COM1` in the Serial Line and `19200` in the Speed, and click Open.
4. Log in using your username and password.
5. At the command line, type `smitty mktcpip` and press Enter.
6. Select network adapter, and press Enter.
7. Enter all IP configuration information.
8. Select Start Now at the bottom of the screen, and press Enter.
9. Press F10 to exit.

Connecting to the IBM POWER5+ processor-based server

We exported the display from the IBM POWER5+ processor-based server to the controller so we could set up programs via a GUI interface.

1. Double-click the Cygwin shortcut on desktop to open the program.
2. At the command prompt, type `startxwin` and press Enter.
3. At command prompt in new window, type `xhost remote_server_IP_address` and press Enter.
4. Type `telnet remote_server_IP_address` and press Enter.
5. At login prompt, enter your username and password.
6. Type `DISPLAY=controller_IP_address` and press Enter.
7. Type `export DISPLAY` and press Enter.

Installing IBM WebSphere on the IBM POWER5+ processor-based server

We used the following steps to install IBM WebSphere Application Server Network Deployment Trial V8.0.

1. On the IBM server at the root directory, type `mkdir websphere` in the command line, and press Enter.
2. Download IBM Installation Manager installed from the IBM site (http://www.ibm.com/developerworks/downloads/ws/wasdevelopers/), and save it to the /websphere directory.
3. At the command line, type `cd /websphere` and press Enter.
4. Type `unzip IBMIM_aix.zip` (where IBMIM_aix.zip is the downloaded installer), and press Enter.
5. Type `./consoleinst.sh` and press Enter.
6. At the Select packages to install prompt, select IBM Installation Manager 1.4.4.
7. Type `n` to continue.
8. At the license agreement prompt, type `a` to accept the agreement.
9. Type `n` to continue.
10. At the Installation Manager installation location, keep the default location.
11. Type `n` to continue.
12. At the install completed successfully prompt, type `r` to continue.
13. At the IBM Installation Manager prompt, type `X` to exit the menu.
14. Type `cd /opt/IBM/InstallationManager/eclipse` and press Enter.
15. Type `./IBMIM` and press Enter.
16. At the IBM Installation Manager main screen, click Install.
17. At the Install Packages screen, select IBM WebSphere Application Server version 8.0.0.1, and click Next.

18. The Install Packages screen should reappear with all fixes and WebSphere Application Server shown. If no additional fixes are found, click Next.
19. At the License Agreement screen, select I accept the terms in the license agreement, and click Next.
20. At Questions for IBM WebSphere Application Server Network Deployment Trial, select the appropriate options for your company, and click Next.
21. At the Select a location for the shared resources directory screen, change the Installation Directory to `/opt/IBM/IMShared` and click Next.
22. At the Install Packages screen, select Create a new Package group.
23. On the same screen, change the Installation Directory to `/opt/WAS8.0/32bit` and click Next.
24. At the Translations screen, select English, and click Next.
25. At the Select the feature to install screen, keep the defaults using IBM 64-bit SDK for Java, and click Next.
26. At the Review the summary information screen, click Install.
27. At the The Packages are installed screen, under Which program do you want to start tab, select None, and click Finish.

Installing IBM DB2 on the IBM POWER5+ processor-based server

We used the steps from the Connecting to the IBM POWER5+ processor-based server section with the database server's IP address before beginning these steps. We used the following steps to install IBM DB2 Data Server 9.7.

1. On the IBM server at the root directory, type `mkdir DB2` at the command line, and press Enter.
2. Download IBM DB2 Data Server 9.7 trial from IBM site (http://www-01.ibm.com/software/data/db2/linux-unix-windows/download.html) to /DB2 directory.
3. At the command line, type `cd /DB2` and press Enter.
4. Type `gzip -c -d v9.7fp4_aix64_server.tar.gz | tar -xvf -` and press Enter. (Where v9.7fp4_aix64_server.tar.gz is the file name from the download.)
5. Type `cd server` and press Enter.
6. Type `./db2setup` and press Enter.
7. At the DB2 Setup Launchpad, click Install a Product.
8. At the Install a Product screen, scroll down to DB2 Enterprise Server Edition Version 9.7 Fix Pack4, and click Install New under this version.
9. At the Welcome to the DB2 Setup Wizard screen, click Next.
10. At the Software License Agreement screen, select Accept, and click Next.
11. At the Select the Installation Type screen, select Typical, and click Next.
12. At the Select installation, response file creation, or both screen, select Install DB2 Enterprise Server Edition on this computer and save my settings in a response file, and click Next using the default Response file name.
13. At the Select the installation directory screen, keep the default location, and click Next.
14. At the Install the IBM Tivoli System Automation for Multiplatforms, select Do not install SA MP, and click Next.
15. At the Set user information for the DB2 Administration Server screen, enter the password for default user, and click Next.
16. At the Set up a DB2 instance screen, keep defaults, and click Next.
17. At the Set up partitioning options for the DB2 instance screen, keep defaults, and click Next.
18. At the Set user information for the DB2 instance owner screen, enter the password for the default user, and click Next.
19. At the Set user information for the fenced owner screen, enter the password for the default user, and click Next.
20. At the Prepare the DB2 tools catalog screen, select Do not prepare the DB2 tools catalog, and click Next.
21. At the Set up notifications screen, select Do not set up your DB2 server to send notifications at this time, and click Next.
22. At the Start copying files and create response file screen, click Finish.
23. At the Setup has completed successfully screen, click Finish.

Setting up the emulator

We installed a fresh installation of SUSE Enterprise Linux 11 as outlined in the Installing SUSE on the Intel E7-8870-based server below. After the installation, we configured the IP addresses as we outlined in Figure 11.

After the installation, make the below configuration changes to /etc/sysctl.conf and /etc/security/limits.conf:

1. Add the following lines to the file /etc/sysctl.conf:

```
fs.file-max = 1048576
kernel.sem = 4096 512000 1600 9000
kernel.shmall = 4294967296
kernel.shmmax = 68719476736
net.core.netdev_max_backlog = 400000
net.core.optmem_max = 30000000
net.core.rmem_default = 30000000
net.core.wmem_default = 30000000
net.core.rmem_max = 30000000
net.core.wmem_max = 30000000
net.core.somaxconn = 300000
net.ipv4.conf.default.rp_filter = 1
net.ipv4.conf.default.accept_source_route = 0
net.ipv4.ip_forward = 0
net.ipv4.ip_local_port_range = 1024 65000
net.ipv4.tcp_fin_timeout = 10
net.ipv4.tcp_max_syn_backlog = 30000
net.ipv4.tcp_max_tw_buckets = 2000000
net.ipv4.tcp_mem  = 30000000 30000000 30000000
net.ipv4.tcp_rmem = 30000000 30000000 30000000
net.ipv4.tcp_wmem = 30000000 30000000 30000000
net.ipv4.tcp_timestamps = 0
net.ipv4.tcp_sack = 0
net.ipv4.conf.all.arp_announce = 2
net.ipv4.conf.all.arp_ignore = 1
```

2. Add the following line to the file /etc/security/limits.conf:
   ```
   *       -   nofile      1048576
   ```
3. Next, install WebSphere as we outline in the Installing WebSphere in SUSE VM on the Intel E7-4870-based server section below.
4. Extract Java application server2010 into the folder /opt/Java_application_server2010.
5. Open command line, type `cd /opt/Java_application_server2010` and press Enter.
6. At the command line, run `ANT_HOME=/opt/Java_application_server2010/ant/apache-ant-1.7.1`
7. At the command line, type `PATH=$ANT_HOME:$PATH` and press Enter.
8. At command line, run `JAVA_HOME=/opt/WAS8.0/32bit/java`
9. In the root directory of the benchmark kit, configure the **build.properties** file. The following properties need to be set for your environment:
   ```
   appserver=websphere
   ```

`database=db2`

10. At the command line, run `ant.install`

Configuring the Java application server2010 configuration files

After setting up the emulator, we edited the following files in the /opt/Java_application_server2010 directory:

Folder appservers/websphere:

- build.properties
- build.xml
- oibm-ejb-jar-bnd.xml
- opersistence.xml

Folder database/db2:

- database.properties

Building Java application server2010 EAR files

After configuring the Java application server2010 files, we ran the following commands:

1. Open the command line, type `cd /opt/Java_application_server2010` and press Enter.
2. At the command line, run `ANT_HOME=/opt/Java_application_server2010/ant/apache-ant-1.7.1`
3. At the command line, type `PATH=$ANT_HOME:$PATH` and press Enter.
4. At the command line, run `JAVA_HOME=/opt/WAS8.0/32bit/java`
5. At the command line, type `ant testj.ear` and press Enter.
6. At the command line, type `ant emulator.ear` and press Enter.
7. Copy testj.ear to both the IBM and Intel WebSphere application server in the /opt directory.
8. Copy emulator.ear file to /opt directory on the emulator.

After these steps, we ran the following script to configure WebSphere on the emulator:

`emulator:/opt/setup_wa8_emulator.sh`

Configuring DB2 on the IBM POWER5+ processor-based server

Before doing the following tasks, the IP addresses should be set as in Figure 13. You should also make sure DNS is setup and configured as in Figure 14. We configured the EMC storage as jfs2 file system for the data volumes. For logging, we used raw volumes without formatting. Run the following script on the database server (see Appendix C for the contents of the script):

`powerdb:/home/db2inst1/createNewTestDB_first.sh`

1. On the Emulator, run the following commands:
    a. Open the command line, type `cd /opt/Java_application_server2010` and press Enter.
    b. At the command line, run `ANT_HOME=/opt/Java_application_server2010/ant/apache-ant-1.7.1`
    c. At the command line, type `PATH=$ANT_HOME:$PATH` and press Enter.
    d. At the command line, run `JAVA_HOME=/opt/WAS8.0/32bit/java`
    e. At the command line, type `ant database.configure` and press Enter.
    f. At the command line, type `ant load.database` and press Enter.
2. Run the following scripts on the database server:
    - `powerdb:/home/db2inst1/createNewTestDB_second.sh`
    - `powerdb:/home/db2inst1/db2tune.sh`

After completing these steps, we backed up the database to one of the data volumes.

Configuring WebSphere on the IBM POWER5+ processor-based server

Before completing the following tasks, the IP addresses should be set in Figure 12. You should also make sure DNS is setup and configured as in Figure 14. We ran the following scripts to configure the server (see Appendix C for the contents of the script).

- `powerapp:/opt/setup_was8_multi.sh`
- `powerapp:/opt/ptwas.py`

After running these scripts, we set the following NIC tuning options on the WebSphere server.

- `rxbuf_pool_sz=4096`
- `rxdesc_que_sz=2048`
- `intr_rate=10000`

Setting up the Intel Xeon processor-based servers

We used the following steps to configure the Intel Xeon processor E7-4870- and E7-8870-based servers for testing.

Installing ESXi 5.0 on the Intel Xeon processor E7-4870 based server

1. Insert the disc, and boot into it.
2. At the ESXi-5.0.0-4695412 standard boot menu, select the installer, and press Enter.
3. At the Welcome to VMware ESXI 5.0.0 Installation screen, press Enter.
4. At the EULA, press F11 to Accept, and Continue.
5. At the Select a Disk to Install or Upgrade screen, select the disk, and press Enter.
6. At the ESX and VMFS found screen, select Install ESXi, preserve VMFS datastore, and press Enter.
7. At the select a keyboard layout screen, press Enter.
8. At the Enter a root password screen, type a password, and press Enter.
9. On the Confirm Install screen, press F11 to install.
10. On the Installation Complete screen, press Enter to reboot.
11. On the ESXi 5.0.0 screen, press F2.
12. At the Authentication Required screen, enter a login name and password, and press Enter.
13. At the System Customization screen, select Configure Management Network, and press Enter.
14. At the Configure Management Network screen, select IP Configuration, and press Enter.
15. At the IP Configuration screen, select Set a static IP, fill in information, and press Enter.
16. On the Configure Management Network screen, press ESC.
17. At the Configure Management Network: Confirm screen, type `Y` to apply changes.
18. At the System Customization screen, press ESC to log out.
19. At the ESX 5.0.0 screen, press F2.
20. At the System Customization screen, select Configure Management Network, and press Enter.
21. At the Configure Management Network screen, select Network Adapters, and press Enter.
22. At the Network Adapters screen, make sure vmnic1 and vmnic2 are connected, and press Enter.
23. At the Configure Management Network screen, press Esc to log out.

Installing SUSE in ESXi5.0 VM on the Intel Xeon processor E7-4870-based server

1. At the vSphere screen, select the WebSphere VM, and click Power on the virtual machine.
2. On the vSphere client screen, select the WebSphere VM, right-click and open Console.
3. At the WebSphere VM console, select the connect/disconnect the CD/DVD devices on the virtual machine.
4. At the Virtual Machine properties screen, select the CD/DVD drive 1.
5. At the Device Type, select host Device, and click Okay.
6. At the SUSE Linux Enterprise Server screen, select Installation.
7. At the Welcome screen, select the language and keyboard layout, check the I agree to the license terms, and click Next.

8. At the Media Check screen, click Next.
9. At the System Probing screen, click Next.
10. At the Installation Mode, select the New Installation, and click Next.
11. At the Clock and Time Zone screen, select the Region and Time Zone, and click Next.
12. At the Server Base Scenario, select Physical Machine(Also for Fully Virtualized Guests), and click Next.
13. At the Installation Settings, review the settings, and click Install.
14. At the YaST2 confirm Package License, click I Agree.
15. At the YaST2 confirm installation screen, click Install.
16. At the Password for the System Administrator root, enter a password and confirm, and click Next.
17. At the Hostname and Domain Name, enter in a username and domain name, check the Change Hostname via DHCP, and click Next.
18. At the Network Configuration screen, select Use Following Configuration, and click Next.
19. At the Network Settings screen, select the listed adapter, and click Edit.
20. At the Network Card Setup, select statically assigned IP Address and fill in the information, and click Next.
21. At the Network settings screen, select the Routing tab and enter in a default gateway, and click OK.
22. At the Network Configuration screen, select Use Following Configuration, and click Next.
23. At the Test Internet Connection screen select No, Skip This Test, and click Next.
24. At the Network Services Configuration screen, select Use Following Configuration, and click Next.
25. At the Network Services Configuration screen, select Skip Configuration, and click Next.
26. At the Network Services Configuration screen, select Use Following Configuration, and click Next.
27. At the User Authentication Method screen, select Local, and click Next.
28. At the New Local User screen, enter a username and Password, and click Next.
29. At the YAST2 Empty User Login screen, click Yes.
30. At the Release Notes screen, click Next.
31. At the Hardware Configuration screen, select Use Following Configuration, and click Next.
32. At the YaST2 prompt about screen flickering, click OK.
33. At the Hardware configuration screen, review the information, and click Next.
34. At the Installation Complete screen, click Finish.

Installing WebSphere in SUSE VM on the Intel Xeon processor E7-4870-based server

1. Open a Terminal, and type `unzip IBMIM_linux-x86.zip`
2. At the terminal, type `ls`
3. At the terminal, type `./install`
4. At the Password Required screen, fill in IBM ID and Password, and click OK.
5. At the Install Packages screen, check IBM installation Manager and IBM WebSphere Application Server Trial, and click check for other versions, fixes, and extensions.
6. At the Install Packages screen, select 8.0.0.1, and click Next.
7. At the Search Results screen, click OK.
8. At the Install Packages screen, click Next.
9. At the install packages license agreement screen, select I accept, and click Next.
10. At the Install Packages Location screen, select the Installation Manager Directory, and click Next.
11. At the Install Packages Summary screen, review the packages selected, and click Install.
12. At the Install Packages Screen, when all the packages are installed, and click Restart Installation Manager.

After the installation, make the following configuration changes to /etc/sysctl.conf and /etc/security/limits.conf:

1. Add the following lines to the file /etc/sysctl.conf:

```
fs.file-max = 1048576
kernel.sem = 250 32000 100 128
kernel.shmall = 4294967296
kernel.shmmax = 68719476736
net.core.netdev_max_backlog = 400000
net.core.optmem_max = 30000000
net.core.rmem_default = 30000000
net.core.wmem_default = 30000000
net.core.rmem_max = 30000000
net.core.wmem_max = 30000000
net.core.somaxconn = 300000
net.ipv4.conf.default.rp_filter = 1
net.ipv4.conf.default.accept_source_route = 0
net.ipv4.ip_forward = 0
net.ipv4.ip_local_port_range = 1024 65000
net.ipv4.tcp_fin_timeout = 10
net.ipv4.tcp_max_syn_backlog = 30000
net.ipv4.tcp_max_tw_buckets = 2000000
net.ipv4.tcp_mem  = 30000000 30000000 30000000
net.ipv4.tcp_rmem = 30000000 30000000 30000000
net.ipv4.tcp_wmem = 30000000 30000000 30000000
net.ipv4.tcp_timestamps = 0
net.ipv4.tcp_sack = 0
net.ipv4.conf.all.arp_announce = 2
net.ipv4.conf.all.arp_ignore = 1
```

2. Add the following line to the file /etc/security/limits.conf:
```
*               -    nofile          1048576
```

Installing SUSE on the Intel Xeon processor E7-8870-based server

1. Boot the server with SUSE Linux Enterprise Server DVD.
2. At the SUSE Linux Enterprise Server screen, select Installation.
3. At the Welcome screen, select the language and keyboard layout, check the I agree to the license terms, and click Next.
4. At the Media Check screen, click Next.
5. At the System Probing screen, click Next.
6. At the Installation Mode, select the New Installation, and click Next.
7. At the Clock and Time Zone screen, select the Region and Time Zone, and click Next.
8. At the Server Base Scenario, select Physical Machine (Also for Fully Virtualized Guests), and click Next.
9. At the Installation Settings, review the settings, and click Install.
10. At the YaST2 confirm Package License, click I Agree.
11. At the YaST2 confirm installation screen, click Install.
12. At the Password for the System Administrator root, enter a password and confirm, and click Next.
13. At the Hostname and Domain Name, enter in a username and domain name, check the Change Hostname via DHCP, and click Next.
14. At the Network Configuration screen, select Use Following Configuration, and click Next.
15. At the Network Settings screen, select the listed adapter, and click Edit.

16. At the Network Card Setup, select statically assigned IP Address and fill in the information, and click Next.
17. At the Network settings screen, select the Routing tab and enter in a default gateway, and click OK.
18. At the Network Configuration screen, select Use Following Configuration, and click Next.
19. At the Test Internet Connection screen select No, Skip This Test, and click Next.
20. At the Network Services Configuration screen, select Use Following Configuration, and click Next.
21. At the Network Services Configuration screen, select Skip Configuration, and click Next.
22. At the Network Services Configuration screen, select Use Following Configuration, and click Next.
23. At the User Authentication Method screen, select Local, and click Next.
24. At the New Local User screen, enter a username and Password, and click Next.
25. At the YAST2 Empty User Login screen, click Yes.
26. At the Release Notes screen, click Next.
27. At the Hardware Configuration screen, select Use Following Configuration, and click Next.
28. At the YaST2 prompt about screen flickering, click OK.
29. At the Hardware configuration screen, review the information, and click Next.
30. At the Installation Complete screen, click Finish.

Installing IBM DB2 on the Intel Xeon processor E7-8870-based server

1. Download IBM DB2 Data Server 9.7 trial from the IBM site (http://www-01.ibm.com/software/data/db2/linux-unix-windows/download.html) to / directory.
2. Type `tar -xvf v9.7fp4_linuxx64_server.tar` and press Enter. (Where v9.7fp4_linuxx64_server.tar is the file name from the download.)
3. Type `cd server` and press Enter.
4. Type `./db2setup` and press Enter.
5. At the DB2 Setup Launchpad, scroll down to DB2 Enterprise Server Edition Version 9.7 Fix Pack 4, and click Install New.
6. At the Welcome to the DB2 Setup Wizard screen, click Next.
7. At the Software License Agreement screen, select Accept, and click Next.
8. At the Select the Installation Type screen, select Typical, and click Next.
9. At the Select installation, response file creation, or both screen, select Install DB2 Enterprise Server Edition on this computer and save my settings in a response file, and click Next using the default Response file name.
10. At the Select the installation directory screen, enter `/opt/ibm/db2/V9.7` and click Next.
11. At the Install the IBM Tivoli System Automation for Multiplatforms, select Do not install SA MP, and click Next.
12. At the Set user information for the DB2 Administration Server screen, enter the password for default user, and click Next.
13. At the Set up a DB2 instance screen, keep defaults, and click Next.
14. At the Set up partitioning options for the DB2 instance screen, keep defaults, and click Next.
15. At the Set user information for the DB2 instance owner screen, enter the password for the default user, and click Next.
16. At the Set user information for the fenced owner screen, enter the password for the default user, and click Next.
17. At the Prepare the DB2 tools catalog screen, select Do not prepare the DB2 tools catalog, and click Next.
18. At the Set up notifications screen, select Do not set up your DB2 server to send notifications at this time, and click Next.
19. At the Start copying files and create response file screen, click Finish.
20. At the Setup has completed successfully screen, click Finish.

After the installation, make the following configuration changes to /etc/sysctl.conf and /etc/security/limits.conf:

1. Add the following lines to the file /etc/sysctl.conf:

```
fs.file-max = 1048576
```

```
kernel.sem = 250 32000 100 128
kernel.shmall = 4294967296
kernel.shmmax = 68719476736
net.core.netdev_max_backlog = 250000
net.core.optmem_max = 30000000
net.core.rmem_default = 30000000
net.core.wmem_default = 30000000
net.core.rmem_max = 32554432
net.core.wmem_max = 32554432
net.core.somaxconn = 32767
net.ipv4.conf.all.arp_announce = 2
net.ipv4.conf.all.arp_ignore = 1
net.ipv4.conf.default.rp_filter = 1
net.ipv4.conf.default.accept_source_route = 0
net.ipv4.ip_forward = 0
net.ipv4.ip_local_port_range = 1024 65000
net.ipv4.tcp_fin_timeout = 30
net.ipv4.tcp_max_syn_backlog = 30000
net.ipv4.tcp_max_tw_buckets = 2000000
net.ipv4.tcp_mem  = 30000000 30000000 30000000
net.ipv4.tcp_rmem = 20480 174760 32554432
net.ipv4.tcp_wmem = 20480 174760 32554432
net.ipv4.tcp_timestamps = 1
net.ipv4.tcp_sack = 1
vm.nr_hugepages = 2048
```

2. Add the following line to the file /etc/security/limits.conf:
```
*               -    nofile          1048576
```

Configuring DB2 on the Intel Xeon processor E7-8870-based server

Before doing the following tasks, the IP addresses should be set as in Figure 13. You should also make sure DNS is setup and configured as in Figure 14. We configured the EMC storage as ext3 file system for the data volumes. For logging, we used raw volumes without formatting. Run the following script on the database server:

- inteldb:/home/db2inst1/createNewTestDB_first.sh

On Emulator, run the following commands.

1. Open a command line, type `cd /opt/Java_application_server2010` and press Enter.
2. At the command line, run ANT_HOME=/opt/Java_application_server2010/ant/apache-ant-1.7.1
3. At the command line, type `PATH=$ANT_HOME:$PATH` and press Enter.
4. At the command line, run JAVA_HOME=/opt/WAS8.0/32bit/java.
5. At the command line, type `ant database.configure` and press Enter.
6. At the command line, type `ant load.database` and press Enter.
7. Run the following scripts on the database server:
   - inteldb:/home/db2inst1/createNewTestDB_second.sh

- inteldb:/home/db2inst1/db2tune.sh

After completing these steps, we backed up the database to one of the data volumes.

Configuring WebSphere on the Intel Xeon processor E7-4870-based server

Before doing the following tasks, the IP addresses should be set as in Figure 12. You should also make sure DNS is setup and configured as in Figure 14. We ran the following scripts to configure the server:

- `inteldmgr:/opt/setup_was8_dmgr.sh`
- `inteldmgr:/opt/ptwas.py`

Setting up the drivers for testing

We installed a fresh installation of SUSE Enterprise Linux 11 as outlined in the Installing SUSE on the Intel Xeon processor E7-8870-based server below. After the installation, we configured the IP addresses as in Figure 10.

After the installation, make the following configuration changes to /etc/sysctl.conf and /etc/security/limits.conf:

1. Add the following lines to the file /etc/sysctl.conf:

```
fs.file-max = 1048576
kernel.sem = 4096 512000 1600 9000
kernel.shmall = 4294967296
kernel.shmmax = 68719476736
net.core.netdev_max_backlog = 400000
net.core.optmem_max = 30000000
net.core.rmem_default = 30000000
net.core.wmem_default = 30000000
net.core.rmem_max = 30000000
net.core.wmem_max = 30000000
net.core.somaxconn = 300000
net.ipv4.conf.default.rp_filter = 1
net.ipv4.conf.default.accept_source_route = 0
net.ipv4.ip_forward = 0
net.ipv4.ip_local_port_range = 1024 65000
net.ipv4.tcp_fin_timeout = 10
net.ipv4.tcp_max_syn_backlog = 30000
net.ipv4.tcp_max_tw_buckets = 2000000
net.ipv4.tcp_mem  = 30000000 30000000 30000000
net.ipv4.tcp_rmem = 30000000 30000000 30000000
net.ipv4.tcp_wmem = 30000000 30000000 30000000
net.ipv4.tcp_timestamps = 0
net.ipv4.tcp_sack = 0
net.ipv4.conf.all.arp_announce = 2
net.ipv4.conf.all.arp_ignore = 1
```

2. Add the following line to the file /etc/security/limits.conf:

```
*                -    nofile          1048576
```

3. Copy the /opt/Java_application_server2010 from the emulator to the same path on all drivers.

Running the benchmark

We used a series of scripts to prepare the benchmark. Run the scripts in the order we present below.

1. Stop all instances of WebSphere on java application server and reboot (/opt/stop_all.sh).
2. Restore the database, and reboot the database server.
3. Reboot the drivers.
4. Stop WebSphere on the emulator, and reboot.
5. After reboot completes on all systems, start WebSphere on the application server and the emulator.
6. On the emulator, run the following command in the /opt/Java_application_server2010 folder:
   ```
   ant faban.harness.start
   ```
7. Open a Web browser on any system and go to the following URL: http://emulator01:9980/.

To start the test, we used the Java_application_server2010 GUI. Before running the test, we filled in the following fields inside the GUI.

Java tab	
JAVA HOME	`/opt/WAS8.0/32bit/java/jre`
JVM Options	`-classpath /opt/Java_application_server2010-1.02/faban/harness/faban/lib/fabancommon.jar:/opt/Java_application_server2010-1.02/faban/harness/faban/lib/fabandriver.jar:/opt/Java_application_server2010-1.02/faban/harness/faban/lib/fabanagents.jar:/opt/Java_application_server2010-1.02/faban/harness/faban/lib/commons-codec-1.2.jar:/opt/Java_application_server2010-1.02/faban/harness/faban/lib/commons-httpclient-3.1.jar:/opt/Java_application_server2010-1.02/target/jar/testjdriver.jar:/opt/Java_application_server2010-1.02/target/jar/testj.jar:/opt/WAS8.0/32bit/endorsed_apis/jaxws-api.jar:/opt/WAS8.0/32bit/runtimes/com.ibm.jaxws.thinclient_8.0.0.jar:/opt/WAS8.0/32bit/runtimes/com.ibm.ws.webservices.thinclient_8.0.0.jar:/opt/WAS8.0/32bit/runtimes/com.ibm.ws.ejb.thinclient_8.0.0.jar:/opt/WAS8.0/32bit/runtimes/com.ibm.ws.jpa.thinclient_8.0.0.jar:/opt/WAS8.0/32bit/runtimes -Xms2048M -Xmx2048M -Djava.net.preferIPv4Stack=true -Dsun.net.inetaddr.ttl=0 -Dnetworkaddress.cache.ttl=0 -Dcom.ibm.websphere.naming.jndicache.cacheobject=none -Dcom.ibm.websphere.naming.hostname.normalizer=...none... -DdisableWSAddressCaching=true -Dcom.ibm.CORBA.ConnectionMultiplicity=25 -Dcom.ibm.CORBA.FragmentSize=10000`

Driver tab	
Host	`driver01 driver02 driver03 driver04`
Stats Collection Interval	`15`
Run time Stats	`True`
Run time Stats Interval	`15`

Scale	`IBM: 500`
	`INTEL: 1300`

Delay between thread starts (ms)	`100`

Ramp Up	1200
Steady State	3600
Ramp Down	300

Dealer Driver tab	
Agents	8

Manufacturing (Mfg) Driver tab	
Agents	4

Measuring power

To record each server's power consumption during each test, we used an Extech Instruments (www.extech.com) 380803 Power Analyzer/Datalogger. We connected the power cord from the server under test to the Power Analyzer's output load power outlet. We then plugged the power cord from the Power Analyzer's input voltage connection into a power outlet.

We used the Power Analyzer's Data Acquisition Software (version 2.11) to capture all recordings. We installed the software on a separate Intel processor-based PC, which we connected to the Power Analyzer via an RS-232 cable. We captured power consumption at one-second intervals.

To gauge the idle power usage, we recorded the power usage for two minutes while each system was running the operating system, but otherwise idle, meaning they were not running any test workload.

We then recorded the power usage (in watts) for each system during the testing at one-second intervals. To compute the average power usage, we averaged the power usage during the time the system was producing its peak performance results. We call this time the power measurement interval. See Figure 5 for the results of these measurements.

APPENDIX C – SCRIPTS USED TO SET UP TEST SERVERS

We used the following scripts to configure the test servers for testing.

powerdb:/home/db2inst1/createNewTestDB_first.sh

```
db2stop force
ipclean -a
db2start

echo "drop DB   "
db2 -v drop db testdb
db2set DB2_APM_PERFORMANCE=
sleep 5
sync
sleep 5
db2stop
sleep 5
sync
sleep 5
dd if=/dev/zero of=/dev/rloglv bs=4096k count=21
sync
sleep 1
db2start
sleep 1
echo "create DB   "
db2 -v create db testdb on /testdata1, /testdata2, /testdata3, /testdata4
#db2 -v create db testdb on /db2_data

echo "Update configuration  "
db2 -v connect to testdb
db2 -v update db cfg for testdb using newlogpath /dev/rloglv
db2 -v update db cfg for testdb using logfilsiz 65535
#db2 -v update db cfg for testdb using LOGPRIMARY 52
db2 -v update database manager configuration using svcename 50000
db2set DB2COMM=TCPIP
db2stop force
db2start

echo "Done!   "
```

powerdb:/home/db2inst1/createNewTestDB_second.sh

```
db2 connect to testdb
echo "PCTFREE ON ORDER TABLES   "
db2 alter table O_ORDERLINE pctfree 99
db2 alter table O_CUSTINVENTORY pctfree 99
db2 alter table O_ORDERS pctfree 99
```

```
echo "Reorgchk   "
db2 reorgchk update statistics
db2 disconnect all

echo "Done!   "
db2stop
db2start
```

powerdb:/home/db2inst1/db2tune.sh

```
PATH=$PATH:/home/db2inst1/sqllib/bin:/home/db2inst1/sqllib/adm:/home/db2inst1/sqllib/misc

ulimit -s unlimited; ulimit -d unlimited; ulimit -m unlimited; ulimit -f unlimited;
ulimit -n unlimited

db2set DB2_USE_ALTERNATE_PAGE_CLEANING=ON
db2set DB2_KEEPTABLELOCK=ON
db2set DB2_APM_PERFORMANCE=ON
db2 -v update dbm cfg using DFT_MON_BUFPOOL on DFT_MON_LOCK off DFT_MON_SORT off
DFT_MON_STMT on DFT_MON_TIMESTAMP on DFT_MON_UOW on  DFT_MON_TABLE off
db2 -v connect to testdb
db2 -v update db cfg for testdb using LOGBUFSZ 4096
db2 -v update db cfg for testdb using SOFTMAX 200
db2 -v update db cfg for testdb using CHNGPGS_THRESH 99
db2 -v update db cfg for testdb using LOGFILSIZ 65535
db2 -v update db cfg for testdb using LOGSECOND 0
#db2 -v update db cfg for testdb using LOGPRIMARY 52
#db2 -v update db cfg for testdb using NEWLOGPATH /dev/testlogs/testlogLV
db2 -v update db cfg for testdb using AUTO_MAINT off
db2 -v update db cfg for testdb using AUTO_RUNSTATS off
db2 -v update db cfg for testdb using AUTO_TBL_MAINT off
db2 -v update db cfg for testdb using LOCKLIST 18000 MAXLOCKS 100
db2 -v update db cfg for testdb using MINCOMMIT 1
db2 -v update db cfg for testdb using MON_REQ_METRICS none
db2 -v update db cfg for testdb using MON_ACT_METRICS none
db2 -v update db cfg for testdb using MON_OBJ_METRICS none
db2 -v alter bufferpool IBMDEFAULTBP immediate size 4000000
db2 -v alter table O_ITEM  volatile
db2 -v alter table M_WORKORDER volatile
db2 -v terminate
db2set DB2COMM=TCPIP
db2stop force
db2start
```

powerapp:/opt/setup_was8_multi.sh

```
#!/bin/sh
```

```
cd /opt/WAS8.0/32bit

DMGR_NAME=Dmgr01
DMGR_HOST=localhost
DMGR_STARTPORT=20000
NODE_HOSTS='powerapp01.test.lan powerapp02.test.lan'

bin/manageprofiles.sh -create \
   -profileName ${DMGR_NAME} \
   -templatePath profileTemplates/management \
   -serverType "DEPLOYMENT_MANAGER" \
   -nodeName ${DMGR_NAME}Node \
   -cellName ${DMGR_NAME}NodeCell \
   -hostName $DMGR_HOST \
   -isDefault \
   -startingPort $DMGR_STARTPORT
cat profiles/${DMGR_NAME}/logs/AboutThisProfile.txt
DMGR_PORT=`awk -F':' '/SOAP/{print $2}' profiles/${DMGR_NAME}/logs/AboutThisProfile.txt`
bin/startManager.sh

n=1
for node_host in $NODE_HOSTS;
do
  node_name="Custom0${n}"
  node_startport=`expr $n \* 1000 + $DMGR_STARTPORT`
  echo "${node_name}: ${node_host}@${node_startport}"
  awk "BEGIN{FS=\"=\";p=${node_startport}}{printf(\"%s=%d\n\",\$1,p++)}"
profileTemplates/managed/actions/portsUpdate/portdef.props >
/tmp/${node_name}_portdef.props
  bin/manageprofiles.sh -create \
     -profileName ${node_name} \
     -templatePath profileTemplates/managed \
     -nodeName ${node_name}Node \
     -cellName ${node_name}Cell \
     -hostName ${node_host} \
     -dmgrHost $DMGR_HOST \
     -dmgrPort $DMGR_PORT \
     -portsFile /tmp/${node_name}_portdef.props \
     -validatePorts
  cat profiles/${node_name}/logs/AboutThisProfile.txt
  n=`expr $n + 1`
done
```

powerapp:/opt/ptway.py

```
#                testNodeName, testServerName, testBusName, testAppHost, testDBHost
testNodeMap = ( ('Custom01Node' , 'server1' , 'Custom01Bus' , 'powerapp01.test.lan',
'dbserver01'),
                ('Custom02Node' , 'server2' , 'Custom02Bus' , 'powerapp02.test.lan',
'dbserver02'),)

#                 testQueue_MDB       testQueue_EIS                testQueue_JMS
testQueue_Name         testQueue_JSD
testQueueMap = ( ('LoaderMDB'        , 'eis/LoaderQueueMDB'       , 'jms/LoaderQueue'
, 'LoaderQueue'        , 'LoaderJSD'       ),
                 ('ReceiveMDB'       , 'eis/ReceiveQueueMDB'      , 'jms/ReceiveQueue'
, 'ReceiveQueue'       , 'ReceiveJSD'      ),
                 ('LargerOrderMDB'   , 'eis/LargeOrderQueueMDB'   , 'jms/LargeOrderQueue'
, 'LargeOrderQueue'    , 'LargeOrderJSD'   ),
                 ('FulfillOrderMDB'  , 'eis/FulfillOrderQueueMDB' ,
'jms/FulfillOrderQueue'  , 'FulfillOrderQueue'  , 'FulfillOrderJSD' ),
                 ('BuyerMDB'         , 'eis/BuyerQueueMDB'        , 'jms/BuyerQueue'
, 'BuyerQueue'         , 'BuyerJSD'        ),
                 ('PurchaseOrderMDB' , 'eis/PurchaseOrderMDB'     ,
'jms/PurchaseOrderQueue' , 'PurchaseOrderQueue' , 'PurchaseOrderJSD'),);

testQueueCFList = ('BuyerQueueConnectionFactory', 'LargeOrderQueueConnectionFactory',
'SupplierQueueConnectionFactory', 'LoaderQueueConnectionFactory',
'FulfillOrderQueueConnectionFactory')

#              testDSName      testDSType   testDSConns
testDSMap = ( ('TESTjNoXADS'     , 'NoXA' , 135 ),
              ('TESTjOrderDS'    , 'XA'   , 5  ),
#              ('TESTjMfgDS'      , 'XA'   , 5  ),
#              ('TESTjSupplierDS' , 'XA'   , 5  ),
              ('TESTjLoaderDS'   , 'XA'   , 15  ),);

testSIBDSName = 'TESTjNoXADS'
#                    testDSProp_name                   testDSProp_type
testDSProp_value
testDSNoXAProps = ( ('webSphereDefaultIsolationLevel' , 'java.lang.Integer' , 2     ),
                    ('resultSetHoldability'           , 'java.lang.Integer' , 1     ),);

testDSXAProps =   ( ('webSphereDefaultIsolationLevel' , 'java.lang.Integer' , 2     ),
                    ('resultSetHoldability'           , 'java.lang.Integer' , 1     ),
                    ('downgradeHoldCursorsUnderXa'    , 'java.lang.Boolean' , 'true'),);

testWASRoot = '/opt/WAS8.0/32bit'
testDBUser = 'db2inst1'
testDBPassword = 'password'
testDBName = 'testdb'
testDBPort = '50000'
testDBJars = testWASRoot + '/db2jars/db2jcc.jar ' + testWASRoot +
'/db2jars/db2jcc_license_cu.jar'
```

```
testStmtCacheSize = 120

#               testPoolName         testPoolMin/Max
testPoolMap = { 'ORB.thread.pool'   : (20,50) ,
                'WebContainer'      : (50,50) ,
                'SIBJMSRAThreadPool': (20,20) ,}

testAppEAR = '/opt/testj.ear'
testAppDB = 'DB2UDB_V97'

testIIOP=2809
testWebPort=9080

testInactivityTimeout = 3500
testJvmMin=2560
testJvmMax=2560
testJvmArgs='-Dcom.ibm.CORBA.FragmentSize=10000 -
Djavax.xml.transform.TransformerFactory=org.apache.xalan.processor.TransformerFactoryImpl
-Djavax.xml.parsers.SAXParserFactory=org.apache.xerces.jaxp.SAXParserFactoryImpl -
Djavax.xml.parsers.DocumentBuilderFactory=org.apache.xerces.jaxp.DocumentBuilderFactoryIm
pl -Xss128k -Xgcpolicy:gencon -Xnoloa -Xdisableexplicitgc -Djava.net.preferIPv4Stack=true
-Dsun.net.inetaddr.ttl=0 -Xtrace:none -Xlp -Xmo500m -Xmn2060m -Xgcthreads8 -
DdisableWSAddressCaching=true'
# -Xaggressive -XtlhPrefetch -Xcodecache32m

testDebug = 'true' # true/false

# Function to show object ID and info
def debug(*params):
   if testDebug == 'true':
      if len(params) > 0:
         for param in params:
            if len(param) > 0:
               print param
      print ''

def showid(objectid):
   if objectid != '':
      debug(objectid,AdminConfig.show(objectid))

print "\nTESTj setup script\n"
# Create J2C auth data
AdminTask.setAdminActiveSecuritySettings('[-
customProperties["com.ibm.websphere.security.JAASAuthData.removeNodeNameGlobal=true"]]')
testDSAuth_ID = AdminTask.createAuthDataEntry('[-alias TESTjDataSourceAuthData -user ' +
testDBUser + ' -password ' + testDBPassword + ' ]')
```

```
showid(testDSAuth_ID)

testTarget = ''

# Set up nodes/servers
for (testNodeName,testServerName,testBusName,testAppHost,testDBHost) in testNodeMap:
   print (testNodeName,testServerName,testBusName,testAppHost,testDBHost)

   testNode_ID = AdminConfig.getid('/Node:' + testNodeName + '/')
   showid(testNode_ID)


   # Locate server id, or create server if it does no exist
   testServer_ID = AdminConfig.getid('/Node:' + testNodeName + '/Server:' +
testServerName + '/' )
   if testServer_ID == '':
      testServer_ID = AdminTask.createApplicationServer(testNodeName, '[-name ' +
testServerName + ' -genUniquePorts true ]')
   debug(testServer_ID)

   # Configure server ports
   testServerPortList = AdminTask.listServerPorts (testServerName, ['-
nodeName',testNodeName]).splitlines()
   for testServerPort in testServerPortList:
      testEndPointName = testServerPort[2:].split()[0]
      AdminTask.modifyServerPort (testServerName, ['-nodeName', testNodeName, '-
endPointName', testEndPointName, '-host', testAppHost, '-modifyShared true'])
   AdminTask.modifyServerPort (testServerName, ['-nodeName', testNodeName, '-endPointName
BOOTSTRAP_ADDRESS', '-host', testAppHost, '-port', testIIOP])
   AdminTask.modifyServerPort (testServerName, ['-nodeName', testNodeName, '-endPointName
WC_defaulthost', '-host', testAppHost, '-port', testWebPort, '-modifyShared true'])
   debug(AdminTask.listServerPorts (testServerName, ['-nodeName',testNodeName]))


   # Create JDBC providers (XA and Non-XA)
   testNoXAProvider_ID = AdminTask.createJDBCProvider('[-scope Node=' + testNodeName + '
-databaseType DB2 -providerType "DB2 Universal JDBC Driver Provider" -implementationType
"Connection pool data source" -name "DB2 Universal JDBC Driver Provider" -classpath [' +
testDBJars + ' ] ]')
   showid(testNoXAProvider_ID)

   testXAProvider_ID = AdminTask.createJDBCProvider('[-scope Node=' + testNodeName + ' -
databaseType DB2 -providerType "DB2 Universal JDBC Driver Provider" -implementationType
"XA data source" -name "DB2 Universal JDBC Driver Provider (XA)" -classpath [' +
testDBJars + ' ] ]')
   showid(testXAProvider_ID)
   # Create data sources
   for (testDSName,testDSType,testDSConns) in testDSMap:
      print (testDSName,testDSType,testDSConns)

      if testDSType == 'NoXA':
         testProvider_ID=testNoXAProvider_ID
         testDSProps = testDSNoXAProps
      else:
         testProvider_ID=testXAProvider_ID
         testDSProps = testDSXAProps


      # Create data source and set attributes
      testDS_ID = AdminTask.createDatasource(testProvider_ID, '[-name ' + testDSName + '
-jndiName jdbc/' + testDSName + ' -description \'TESTj Datasource: ' + testDSName + '\' -
dataStoreHelperClassName com.ibm.websphere.rsadapter.DB2UniversalDataStoreHelper -
containerManagedPersistence true -componentManagedAuthenticationAlias
TESTjDataSourceAuthData -configureResourceProperties [[databaseName java.lang.String ' +
testDBName + '] [driverType java.lang.Integer 4] [serverName java.lang.String ' +
testDBHost + '] [portNumber java.lang.Integer ' + testDBPort + '] ] ]')

      AdminConfig.modify(testDS_ID,[['statementCacheSize', testStmtCacheSize]])
      showid(testDS_ID)


      # Setup auth mapping and CMP settings for data source
      testDS_MAP_ID = AdminConfig.create('MappingModule', testDS_ID, '[[authDataAlias
TESTjDataSourceAuthData] [mappingConfigAlias DefaultPrincipalMapping]]')
      showid(testDS_MAP_ID)

      testCMPList = AdminConfig.list('CMPConnectorFactory',testNode_ID).splitlines()
      for testCMP_ID in testCMPList:
         testCMP_DS_ID = AdminConfig.showAttribute(testCMP_ID, 'cmpDatasource')
         if testDS_ID == testCMP_DS_ID:
            break


      AdminConfig.modify(testCMP_ID, '[[authDataAlias "TESTjDataSourceAuthData"]]')
      showid(testCMP_ID)

      testCMP_MAP_ID = AdminConfig.create('MappingModule', testCMP_ID, '[[authDataAlias
TESTjDataSourceAuthData] [mappingConfigAlias DefaultPrincipalMapping]]')
      showid(testCMP_MAP_ID)
      # Modify data source connection pool settings
      testDSPool_ID = AdminConfig.showAttribute(testDS_ID,'connectionPool')
      AdminConfig.modify(testDSPool_ID, [['connectionTimeout', 0], ['maxConnections', 
testDSConns], ['minConnections', testDSConns], ['reapTime', 0], ['unusedTimeout', 0], 
['agedTimeout', 0]])
      showid(testDSPool_ID)


      # Create/modify data source properties
      testDSPropSet_ID = AdminConfig.showAttribute(testDS_ID,'propertySet')
      debug(testDSPropSet_ID)

      for (testDSProp_name,testDSProp_type,testDSProp_value) in testDSProps:
         print (testDSProp_name,testDSProp_type,testDSProp_value)
         testDSProp_attrs = [['name', testDSProp_name], ['type', testDSProp_type], 
['value', testDSProp_value]]
         testDSProp_ID = AdminConfig.getid('/Node:' + testNodeName + 
'/JDBCProvider:/DataSource:' + testDSName + 
'/J2EEResourcePropertySet:/J2EEResourceProperty:' + testDSProp_name + '/')
         if testDSProp_ID == '':
            testDSProp_ID = AdminConfig.create('J2EEResourceProperty', testDSPropSet_ID, 
testDSProp_attrs)
         else:
            AdminConfig.modify(testDSProp_ID, testDSProp_attrs)
         showid(testDSProp_ID)


   # Setup SIBus
   testSIBus_ID = AdminTask.createSIBus('[-bus ' + testBusName + ' -busSecurity false ]')
   showid(testSIBus_ID)

   AdminTask.addSIBusMember('[-bus ' + testBusName + ' -node ' + testNodeName + ' -server 
' + testServerName + ' -dataStore  -createDefaultDatasource false -datasourceJndiName 
jdbc/' + testSIBDSName + ' -authAlias TESTjDataSourceAuthData -createTables true -
schemaName ' + testBusName + ' ]')


   # Create connection factories
   for testQueueCFName in testQueueCFList:
      testJMS_CF_ID = AdminTask.createSIBJMSConnectionFactory(testNode_ID, '[-type queue 
-name ' + testQueueCFName + ' -jndiName jms/' + testQueueCFName + ' -busName ' + 
testBusName + ' -mappingAlias DefaultPrincipalMapping -shareDataSourceWithCMP true -
persistentMapping ReliablePersistent]')
      showid(testJMS_CF_ID)
   # Create SIBus destinations, JMS queues, and MBeans
   for (testQueue_MDB, testQueue_EIS, testQueue_JMS, testQueue_Name, testQueue_JSD) in
testQueueMap:
      print (testQueue_MDB, testQueue_EIS, testQueue_JMS, testQueue_Name, testQueue_JSD)

      testQueue_JSD_ID = AdminTask.createSIBDestination('[-bus ' + testBusName + ' -name
' + testQueue_JSD + ' -node ' + testNodeName + ' -server ' + testServerName + ' -type
Queue -reliability ASSURED_PERSISTENT ]')
      showid(testQueue_JSD_ID)

      testQueue_JMS_ID = AdminTask.createSIBJMSQueue(testNode_ID, '[-name ' +
testQueue_Name + ' -jndiName ' + testQueue_JMS + ' -busName ' + testBusName + ' -
queueName ' + testQueue_JSD + ' -deliveryMode Persistent ]')
      showid(testQueue_JMS_ID)

      testQueue_MDB_ID = AdminTask.createSIBJMSActivationTest(testNode_ID, '[-name ' +
testQueue_MDB + ' -jndiName ' + testQueue_EIS + ' -destinationJndiName ' + testQueue_JMS
+ ' -busName ' + testBusName + ' -destinationType javax.jms.Queue -shareDataSourceWithCMP
true]')
      showid(testQueue_MDB_ID)

   debug("\n\nServer Tuning:")
   # Server Tuning
   AdminConfig.modify(testServer_ID,'[[parallelStartEnabled true]]')
   showid(testServer_ID)

   # ORB/ORB thread pool
   testORB_ID = AdminConfig.list('ObjectRequestBroker',testServer_ID)
   AdminConfig.modify(testORB_ID,'[[noLocalCopies false] [useServerThreadPool true]]')
   showid(testORB_ID)

   testORBPool_ID = AdminConfig.showAttribute(testORB_ID,'threadPool')
   AdminConfig.modify(testORBPool_ID,'[[minimumSize 10] [maximumSize 10]]')
   showid(testORBPool_ID)

   # ORB transport
   testORBMultiHome_ID = AdminConfig.getid('/Node:' + testNodeName + '/Server:' +
testServerName + '/ObjectRequestBroker:/Property:com.ibm.ws.orb.transport.useMultiHome/')
   AdminConfig.modify(testORBMultiHome_ID,'[[value false]]')
   showid(testORBMultiHome_ID)
   testORBLocalhost_ID = AdminConfig.getid('/Node:' + testNodeName + '/Server:' +
testServerName + '/ObjectRequestBroker:/Property:com.ibm.CORBA.LocalHost/')
   if testORBLocalhost_ID != '':
      AdminConfig.modify(testORBLocalhost_ID,[['value',testAppHost]])
   else:
      testORBLocalhost_ID = AdminConfig.create('Property',testORB_ID,'[[name 
com.ibm.CORBA.LocalHost] [value ' + testAppHost + ']]')
   showid(testORBLocalhost_ID)

   # All thread pools
   testPoolList = AdminConfig.list('ThreadPool',testServer_ID).splitlines()
   for testPool_ID in testPoolList:
      AdminConfig.modify(testPool_ID,[['inactivityTimeout',testInactivityTimeout]])
      showid(testPool_ID)

   # Testific thread pools
   for testPoolName,(testPoolMin,testPoolMax) in testPoolMap.items():
      testPool_ID = AdminConfig.getid('/Node:' + testNodeName + '/Server:' + 
testServerName + '/ThreadPoolManager:/ThreadPool:' + testPoolName + '/')
      if testPool_ID != '':
         AdminConfig.modify(testPool_ID,[['minimumSize', testPoolMin], ['maximumSize', 
testPoolMax]])
         showid(testPool_ID)

   # HTTP_2 inbound channel
   testHTTPInbound_ID = AdminConfig.getid('/Node:' + testNodeName + '/Server:' + 
testServerName + '/TransportChannelService:/HTTPInboundChannel:HTTP_2/')
   AdminConfig.modify(testHTTPInbound_ID,'[[maximumPersistentRequests -1] [keepAlive 
true] [readTimeout 6000] [writeTimeout 6000] [persistentTimeout 3000] [enableLogging 
false]]')
   showid(testHTTPInbound_ID)

   # Transaction service
   testTransService_ID = AdminConfig.list('TransactionService',testServer_ID)
   AdminConfig.modify(testTransService_ID,'[[totalTranLifetimeTimeout 0] 
[clientInactivityTimeout 0] [propogatedOrBMTTranLifetimeTimeout 0]]')
   showid(testTransService_ID)

   # JVM tuning
   testJVM_ID = AdminConfig.list('JavaVirtualMachine',testServer_ID)
   AdminConfig.modify(testJVM_ID,[['initialHeapSize', testJvmMin], ['maximumHeapSize', 
testJvmMax], ['genericJvmArguments', testJvmArgs]])
   showid(testJVM_ID)
   testJVMMultiHome_ID = AdminConfig.getid('/Node:' + testNodeName + '/Server:' + 
testServerName + 
'/JavaProcessDef:/JavaVirtualMachine:/Property:com.ibm.websphere.network.useMultiHome/')
   if testJVMMultiHome_ID != '':
      AdminConfig.modify(testJVMMultiHome_ID,[['value',testAppHost]])
   else:
      testJVMMultiHome_ID = AdminConfig.create('Property',testJVM_ID,'[[name 
com.ibm.websphere.network.useMultiHome] [value false]]')
```

```
   showid(testJVMMultiHome_ID)

   # Log tuning
   testLogList = AdminConfig.list('StreamRedirect',testServer_ID).splitlines()
   for testLog_ID in testLogList:
      AdminConfig.modify(testLog_ID,'[[rolloverType NONE] [maxNumberOfBackupFiles 1]]')
      showid(testLog_ID)

   # Trace service/trace log
   testTraceService_ID = AdminConfig.list('TraceService',testServer_ID)
   #AdminConfig.modify(testTraceService_ID,'[[enable false] [startupTraceTestification
"*=fatal"]]')
   #showid(testTraceService_ID)

   testTraceLog_ID = AdminConfig.showAttribute(testTraceService_ID,'traceLog')
   AdminConfig.modify(testTraceLog_ID,'[[rolloverSize 200] [maxNumberOfBackupFiles 20]]')
   showid(testTraceLog_ID)

   # PMI service
   testPMIService_ID = AdminConfig.list('PMIService',testServer_ID)
   AdminConfig.modify(testPMIService_ID,'[[enable false]]')
   showid(testPMIService_ID)

   # EJB cache
   testEJBCache_ID = AdminConfig.list('EJBCache',testServer_ID)
   AdminConfig.modify(testEJBCache_ID,'[[cacheSize 10000]]')

   # Setup install target
   if testTarget != '':
      testTarget += '+'
   testTarget += 'WebSphere:node=' + testNodeName + ',server=' + testServerName

# Install testj App
debug("\nInstall app to target: ",testTarget)
AdminApp.install(testAppEAR,'[-verbose -deployejb -deployejb.dbtype ' + testAppDB + ' -
deployws -usedefaultbindings -useMetaDataFromBinary yes -target ' + testTarget + ' ]')

import code; code.interact(local=locals())

#AdminApp.install('/opt/testj.ear','[-verbose -deployejb -deployejb.dbtype DB2UDB_V97 -
deployws -usedefaultbindings -useMetaDataFromBinary yes -target
WebSphere:node=Custom01Node,server=server1+WebSphere:node=Custom02Node,server=server2 ]')
```

inteldb:/home/db2inst1/createNewTestDB_first.sh

```
db2stop force
ipclean -a
```

```
db2start

echo "drop DB   "
db2 -v drop db testdb
db2set DB2_APM_PERFORMANCE=
sleep 5
sync
sleep 5
db2stop
sleep 5
sync
sleep 5
dd if=/dev/zero of=/dev/logs/LV bs=4096k count=21
sync
sleep 1
db2start
sleep 1
echo "create DB   "
db2 -v create db testdb on /testdata1, /testdata2, /testdata3, /testdata4
#db2 -v create db testdb on /db2_data

echo "Update configuration  "
db2 -v connect to testdb
db2 -v update db cfg for testdb using newlogpath /dev/logs/LV
db2 -v update db cfg for testdb using logfilsiz 65535
#db2 -v update db cfg for testdb using LOGPRIMARY 52
db2 -v update database manager configuration using svcename 50000
db2set DB2COMM=TCPIP
db2stop force
db2start

echo "Done!   "
```

inteldb:/home/db2inst1/createNewTestDB_second.sh

```
db2 connect to testdb
echo "PCTFREE ON ORDER TABLES   "
db2 alter table O_ORDERLINE pctfree 99
db2 alter table O_CUSTINVENTORY pctfree 99
db2 alter table O_ORDERS pctfree 99

echo "Reorgchk   "
db2 reorgchk update statistics
db2 disconnect all

echo "Done!   "
db2stop
```

```
db2start
```

inteldb:/home/db2inst1/db2tune.sh

```
db2set DB2_USE_ALTERNATE_PAGE_CLEANING=ON
db2set DB2_KEEPTABLELOCK=ON
db2set DB2_APM_PERFORMANCE=ON
db2 -v update dbm cfg using DFT_MON_BUFPOOL on DFT_MON_LOCK off DFT_MON_SORT off
DFT_MON_STMT on DFT_MON_TIMESTAMP on DFT_MON_UOW on  DFT_MON_TABLE off
db2 -v connect to testdb
db2 -v update db cfg for testdb using LOGBUFSZ 4096
db2 -v update db cfg for testdb using SOFTMAX 200
db2 -v update db cfg for testdb using CHNGPGS_THRESH 99
db2 -v update db cfg for testdb using LOGFILSIZ 65535
db2 -v update db cfg for testdb using LOGSECOND 0
db2 -v update db cfg for testdb using LOGPRIMARY 150
#db2 -v update db cfg for testdb using NEWLOGPATH /dev/logs/LV
db2 -v update db cfg for testdb using AUTO_MAINT off
db2 -v update db cfg for testdb using AUTO_RUNSTATS off
db2 -v update db cfg for testdb using AUTO_TBL_MAINT off
db2 -v update db cfg for testdb using LOCKLIST 18000 MAXLOCKS 100
db2 -v update db cfg for testdb using MINCOMMIT 1
db2 -v update db cfg for testdb using MON_REQ_METRICS none
db2 -v update db cfg for testdb using MON_ACT_METRICS none
db2 -v update db cfg for testdb using MON_OBJ_METRICS none
#db2 -v alter bufferpool IBMDEFAULTBP immediate size 7000000
db2 -v alter bufferpool IBMDEFAULTBP immediate size 19000000
db2 -v alter table O_ITEM  volatile
db2 -v alter table M_WORKORDER volatile
db2 -v terminate
db2set DB2COMM=TCPIP
db2stop force
db2start
```

emulator:/opt/setup_wa8_emulator.sh

```
#!/bin/sh
cd /opt/WAS8.0/32bit
EMULATOR_HOST=emulator.test.lan

bin/manageprofiles.sh -create \
   -profileName Default01 \
   -templatePath profileTemplates/default \
   -nodeName Default01Node \
   -cellName Default01Cell \
   -hostName $EMULATOR_HOST \
   -omitAction defaultAppDeployAndConfig \
   -isDefault
bin/startServer.sh server1
```

inteldmgr:/opt/setup_was8_dmgr.sh

```
#!/bin/sh
cd /opt/WAS8.0/32bit

DMGR_NAME=Dmgr01
DMGR_HOST=inteldmgr.test.lan
DMGR_STARTPORT=20000
NODE_HOSTS='intelapp01.test.lan intelapp02.test.lan'

bin/manageprofiles.sh -create \
   -profileName ${DMGR_NAME} \
   -templatePath profileTemplates/management \
   -serverType "DEPLOYMENT_MANAGER" \
   -nodeName ${DMGR_NAME}Node \
   -cellName ${DMGR_NAME}NodeCell \
   -hostName $DMGR_HOST \
   -isDefault \
   -startingPort $DMGR_STARTPORT
cat profiles/${DMGR_NAME}/logs/AboutThisProfile.txt
DMGR_PORT=`awk -F':' '/SOAP/{print $2}' profiles/${DMGR_NAME}/logs/AboutThisProfile.txt`
bin/startManager.sh

n=1
for node_host in $NODE_HOSTS;
do
  node_name="Custom0${n}"
  node_startport=`expr $n \* 1000 + $DMGR_STARTPORT`
  echo "${node_name}: ${node_host}@${node_startport}"
#awk "BEGIN{FS=\"=\";p=${node_startport}}{printf(\"%s=%d\n\",\$1,p++)}"
profileTemplates/managed/actions/portsUpdate/portdef.props >
/tmp/${node_name}_portdef.props
ssh $node_host "cd /opt/WAS8.0/32bit ;
awk \"BEGIN{FS=\\\"=\\\";p=${node_startport}}{printf(\\\"%s=%d\n\\\",\\\$1,p++)}\"
profileTemplates/managed/actions/portsUpdate/portdef.props >
/tmp/${node_name}_portdef.props ;
     bin/manageprofiles.sh -create \
     -profileName ${node_name} \
     -templatePath profileTemplates/managed \
     -nodeName ${node_name}Node \
     -cellName ${node_name}Cell \
     -hostName ${node_host} \
     -dmgrHost $DMGR_HOST \
     -dmgrPort $DMGR_PORT \
     -portsFile /tmp/${node_name}_portdef.props \
     -validatePorts ;
  cat profiles/${node_name}/logs/AboutThisProfile.txt"
  n=`expr $n + 1`
done
```

inteldmgr:/opt/ptwas.py

```
#                testNodeName, testServerName, testBusName, testAppHost, testDBHost
testNodeMap = ( ('Custom01Node' , 'server1' , 'Custom01Bus' , 'intelapp01.test.lan',
'dbserver01'),
                ('Custom02Node' , 'server2' , 'Custom02Bus' , 'intelapp02.test.lan',
'dbserver02'),
                ('Custom03Node' , 'server3' , 'Custom03Bus' , 'intelapp03.test.lan',
'dbserver03'),
                ('Custom04Node' , 'server4' , 'Custom04Bus' , 'intelapp04.test.lan',
'dbserver04'),)

#                 testQueue_MDB       testQueue_EIS                 testQueue_JMS
testQueue_Name         testQueue_JSD
testQueueMap = ( ('LoaderMDB'        , 'eis/LoaderQueueMDB'       , 'jms/LoaderQueue'
, 'LoaderQueue'        , 'LoaderJSD'       ),
                 ('ReceiveMDB'       , 'eis/ReceiveQueueMDB'      , 'jms/ReceiveQueue'
, 'ReceiveQueue'       , 'ReceiveJSD'      ),
                 ('LargerOrderMDB'   , 'eis/LargeOrderQueueMDB'   , 'jms/LargeOrderQueue'
, 'LargeOrderQueue'    , 'LargeOrderJSD'   ),
                 ('FulfillOrderMDB'  , 'eis/FulfillOrderQueueMDB' ,
'jms/FulfillOrderQueue'  , 'FulfillOrderQueue'  , 'FulfillOrderJSD' ),
                 ('BuyerMDB'         , 'eis/BuyerQueueMDB'        , 'jms/BuyerQueue'
, 'BuyerQueue'         , 'BuyerJSD'        ),
                 ('PurchaseOrderMDB' , 'eis/PurchaseOrderMDB'     ,
'jms/PurchaseOrderQueue' , 'PurchaseOrderQueue' , 'PurchaseOrderJSD'),);

testQueueCFList = ('BuyerQueueConnectionFactory', 'LargeOrderQueueConnectionFactory',
'SupplierQueueConnectionFactory', 'LoaderQueueConnectionFactory',
'FulfillOrderQueueConnectionFactory')

#              testDSName      testDSType   testDSConns
testDSMap = (  ('TESTjNoXADS'     , 'NoXA' , 135 ),
               ('TESTjOrderDS'    , 'XA'   , 5  ),
#               ('TESTjMfgDS'      , 'XA'   , 5  ),
#               ('TESTjSupplierDS' , 'XA'   , 5  ),
               ('TESTjLoaderDS'   , 'XA'   , 15  ),);

testSIBDSName = 'TESTjNoXADS'
#                    testDSProp_name                    testDSProp_type
testDSProp_value
```

```
testDSNoXAProps = ( ('webSphereDefaultIsolationLevel' , 'java.lang.Integer' , 2     ),
                    ('resultSetHoldability'           , 'java.lang.Integer' , 1     ),);

testDSXAProps =   ( ('webSphereDefaultIsolationLevel' , 'java.lang.Integer' , 2     ),
                    ('resultSetHoldability'           , 'java.lang.Integer' , 1     ),
                    ('downgradeHoldCursorsUnderXa'    , 'java.lang.Boolean' , 'true'),);

testWASRoot = '/opt/WAS8.0/32bit'
testDBUser = 'db2inst1'
testDBPassword = 'password'
testDBName = 'testdb'
testDBPort = '50000'
testDBJars = testWASRoot + '/db2jars/db2jcc.jar ' + testWASRoot +
'/db2jars/db2jcc_license_cu.jar'
testStmtCacheSize = 120

#               testPoolName         testPoolMin/Max
testPoolMap = { 'ORB.thread.pool'    : (20,50) ,
                'WebContainer'       : (50,50) ,
                'SIBJMSRAThreadPool': (20,20) ,}

testAppEAR = '/opt/testj.ear'
testAppDB = 'DB2UDB_V97'

testIIOP=2809
testWebPort=9080
testMultiHome='true'

testInactivityTimeout = 3500
testJvmMin=2560
testJvmMax=2560
testJvmArgs='-Dcom.ibm.CORBA.FragmentSize=10000 -
Djavax.xml.transform.TransformerFactory=org.apache.xalan.processor.TransformerFactoryImpl
-Djavax.xml.parsers.SAXParserFactory=org.apache.xerces.jaxp.SAXParserFactoryImpl -
Djavax.xml.parsers.DocumentBuilderFactory=org.apache.xerces.jaxp.DocumentBuilderFactoryIm
pl -Xss128k -Xgcpolicy:gencon -Xnoloa -Xdisableexplicitgc -Djava.net.preferIPv4Stack=true
-Dsun.net.inetaddr.ttl=0 -Xtrace:none -Xlp -Xmo500m -Xmn2060m -Xgcthreads8 -
DdisableWSAddressCaching=true'

testDebug = 'true' # true/false

# Function to show object ID and info
def debug(*params):
   if testDebug == 'true':
      if len(params) > 0:
         for param in params:
            if len(param) > 0:
               print param
      print ''

def showid(objectid):
   if objectid != '':
      debug(objectid,AdminConfig.show(objectid))

print "\nTESTj setup script\n"
# Create J2C auth data
AdminTask.setAdminActiveSecuritySettings('[-
customProperties["com.ibm.websphere.security.JAASAuthData.removeNodeNameGlobal=true"]]')
testDSAuth_ID = AdminTask.createAuthDataEntry('[-alias TESTjDataSourceAuthData -user ' +
testDBUser + ' -password ' + testDBPassword + ' ]')
showid(testDSAuth_ID)

testTarget = ''

# Set up nodes/servers
for (testNodeName,testServerName,testBusName,testAppHost,testDBHost) in testNodeMap:
   print (testNodeName,testServerName,testBusName,testAppHost,testDBHost)

   testNode_ID = AdminConfig.getid('/Node:' + testNodeName + '/')
   showid(testNode_ID)


   # Locate server id, or create server if it does no exist
   testServer_ID = AdminConfig.getid('/Node:' + testNodeName + '/Server:' +
testServerName + '/' )
   if testServer_ID == '':
      testServer_ID = AdminTask.createApplicationServer(testNodeName, '[-name ' +
testServerName + ' -genUniquePorts true ]')
   debug(testServer_ID)

   # Configure server ports
   if testMultiHome == 'false':
      testServerPortList = AdminTask.listServerPorts (testServerName, ['-
nodeName',testNodeName]).splitlines()
      for testServerPort in testServerPortList:
         testEndPointName = testServerPort[2:].split()[0]
         AdminTask.modifyServerPort (testServerName, ['-nodeName', testNodeName, '-
endPointName', testEndPointName, '-host', testAppHost, '-modifyShared true'])
   AdminTask.modifyServerPort (testServerName, ['-nodeName', testNodeName, '-endPointName
BOOTSTRAP_ADDRESS', '-port', testIIOP])
   AdminTask.modifyServerPort (testServerName, ['-nodeName', testNodeName, '-endPointName
WC_defaulthost', '-port', testWebPort, '-modifyShared true'])
```

```
   debug(AdminTask.listServerPorts (testServerName, ['-nodeName',testNodeName]))


   # Create JDBC providers (XA and Non-XA)
   testNoXAProvider_ID = AdminTask.createJDBCProvider('[-scope Node=' + testNodeName + '
-databaseType DB2 -providerType "DB2 Universal JDBC Driver Provider" -implementationType
"Connection pool data source" -name "DB2 Universal JDBC Driver Provider" -classpath [' +
testDBJars + ' ] ]')
   showid(testNoXAProvider_ID)

   testXAProvider_ID = AdminTask.createJDBCProvider('[-scope Node=' + testNodeName + ' -
databaseType DB2 -providerType "DB2 Universal JDBC Driver Provider" -implementationType
"XA data source" -name "DB2 Universal JDBC Driver Provider (XA)" -classpath [' +
testDBJars + ' ] ]')
   showid(testXAProvider_ID)


   # Create data sources
   for (testDSName,testDSType,testDSConns) in testDSMap:
      print (testDSName,testDSType,testDSConns)

      if testDSType == 'NoXA':
         testProvider_ID=testNoXAProvider_ID
         testDSProps = testDSNoXAProps
      else:
         testProvider_ID=testXAProvider_ID
         testDSProps = testDSXAProps


      # Create data source and set attributes
      testDS_ID = AdminTask.createDatasource(testProvider_ID, '[-name ' + testDSName + '
-jndiName jdbc/' + testDSName + ' -description \'TESTj Datasource: ' + testDSName + '\' -
dataStoreHelperClassName com.ibm.websphere.rsadapter.DB2UniversalDataStoreHelper -
containerManagedPersistence true -componentManagedAuthenticationAlias
TESTjDataSourceAuthData -configureResourceProperties [[databaseName java.lang.String ' +
testDBName + '] [driverType java.lang.Integer 4] [serverName java.lang.String ' +
testDBHost + '] [portNumber java.lang.Integer ' + testDBPort + '] ] ]')

      AdminConfig.modify(testDS_ID,[['statementCacheSize', testStmtCacheSize]])
      showid(testDS_ID)


      # Setup auth mapping and CMP settings for data source
      testDS_MAP_ID = AdminConfig.create('MappingModule', testDS_ID, '[[authDataAlias
TESTjDataSourceAuthData] [mappingConfigAlias DefaultPrincipalMapping]]')
      showid(testDS_MAP_ID)
      testCMPList = AdminConfig.list('CMPConnectorFactory',testNode_ID).splitlines()
      for testCMP_ID in testCMPList:
         testCMP_DS_ID = AdminConfig.showAttribute(testCMP_ID, 'cmpDatasource')
         if testDS_ID == testCMP_DS_ID:
            break


      AdminConfig.modify(testCMP_ID, '[[authDataAlias "TESTjDataSourceAuthData"]]')
      showid(testCMP_ID)

      testCMP_MAP_ID = AdminConfig.create('MappingModule', testCMP_ID, '[[authDataAlias 
TESTjDataSourceAuthData] [mappingConfigAlias DefaultPrincipalMapping]]')
      showid(testCMP_MAP_ID)


      # Modify data source connection pool settings
      testDSPool_ID = AdminConfig.showAttribute(testDS_ID,'connectionPool')
      AdminConfig.modify(testDSPool_ID, [['connectionTimeout', 0], ['maxConnections', 
testDSConns], ['minConnections', testDSConns], ['reapTime', 0], ['unusedTimeout', 0], 
['agedTimeout', 0]])
      showid(testDSPool_ID)


      # Create/modify data source properties
      testDSPropSet_ID = AdminConfig.showAttribute(testDS_ID,'propertySet')
      debug(testDSPropSet_ID)

      for (testDSProp_name,testDSProp_type,testDSProp_value) in testDSProps:
         print (testDSProp_name,testDSProp_type,testDSProp_value)
         testDSProp_attrs = [['name', testDSProp_name], ['type', testDSProp_type], 
['value', testDSProp_value]]
         testDSProp_ID = AdminConfig.getid('/Node:' + testNodeName + 
'/JDBCProvider:/DataSource:' + testDSName + 
'/J2EEResourcePropertySet:/J2EEResourceProperty:' + testDSProp_name + '/')
         if testDSProp_ID == '':
            testDSProp_ID = AdminConfig.create('J2EEResourceProperty', testDSPropSet_ID, 
testDSProp_attrs)
         else:
            AdminConfig.modify(testDSProp_ID, testDSProp_attrs)
         showid(testDSProp_ID)


   # Setup SIBus
   testSIBus_ID = AdminTask.createSIBus('[-bus ' + testBusName + ' -busSecurity false ]')
   showid(testSIBus_ID)
   AdminTask.addSIBusMember('[-bus ' + testBusName + ' -node ' + testNodeName + ' -server
' + testServerName + ' -dataStore  -createDefaultDatasource false -datasourceJndiName
jdbc/' + testSIBDSName + ' -authAlias TESTjDataSourceAuthData -createTables true -
schemaName ' + testBusName + ' ]')


   # Create connection factories
   for testQueueCFName in testQueueCFList:
      testJMS_CF_ID = AdminTask.createSIBJMSConnectionFactory(testNode_ID, '[-type queue
-name ' + testQueueCFName + ' -jndiName jms/' + testQueueCFName + ' -busName ' +
testBusName + ' -mappingAlias DefaultPrincipalMapping -shareDataSourceWithCMP true -
persistentMapping ReliablePersistent]')
      showid(testJMS_CF_ID)


   # Create SIBus destinations, JMS queues, and MBeans
   for (testQueue_MDB, testQueue_EIS, testQueue_JMS, testQueue_Name, testQueue_JSD) in
testQueueMap:
      print (testQueue_MDB, testQueue_EIS, testQueue_JMS, testQueue_Name, testQueue_JSD)

      testQueue_JSD_ID = AdminTask.createSIBDestination('[-bus ' + testBusName + ' -name
' + testQueue_JSD + ' -node ' + testNodeName + ' -server ' + testServerName + ' -type
Queue -reliability ASSURED_PERSISTENT ]')
      showid(testQueue_JSD_ID)

      testQueue_JMS_ID = AdminTask.createSIBJMSQueue(testNode_ID, '[-name ' +
testQueue_Name + ' -jndiName ' + testQueue_JMS + ' -busName ' + testBusName + ' -
queueName ' + testQueue_JSD + ' -deliveryMode Persistent ]')
      showid(testQueue_JMS_ID)

      testQueue_MDB_ID = AdminTask.createSIBJMSActivationTest(testNode_ID, '[-name ' +
testQueue_MDB + ' -jndiName ' + testQueue_EIS + ' -destinationJndiName ' + testQueue_JMS
+ ' -busName ' + testBusName + ' -destinationType javax.jms.Queue -shareDataSourceWithCMP
true]')
      showid(testQueue_MDB_ID)

   debug("\n\nServer Tuning:")
   # Server Tuning
   AdminConfig.modify(testServer_ID,'[[parallelStartEnabled true]]')
   showid(testServer_ID)

   # ORB/ORB thread pool
   testORB_ID = AdminConfig.list('ObjectRequestBroker',testServer_ID)
   AdminConfig.modify(testORB_ID,'[[noLocalCopies false] [useServerThreadPool true]]')
   showid(testORB_ID)
   testORBPool_ID = AdminConfig.showAttribute(testORB_ID,'threadPool')
   AdminConfig.modify(testORBPool_ID,'[[minimumSize 10] [maximumSize 10]]')
   showid(testORBPool_ID)

   # ORB transport
   if testMultiHome == 'false':
      testORBMultiHome_ID = AdminConfig.getid('/Node:' + testNodeName + '/Server:' +
testServerName + '/ObjectRequestBroker:/Property:com.ibm.ws.orb.transport.useMultiHome/')
      AdminConfig.modify(testORBMultiHome_ID,'[[value false]]')
      showid(testORBMultiHome_ID)
      testORBLocalhost_ID = AdminConfig.getid('/Node:' + testNodeName + '/Server:' +
testServerName + '/ObjectRequestBroker:/Property:com.ibm.CORBA.LocalHost/')
      if testORBLocalhost_ID != '':
         AdminConfig.modify(testORBLocalhost_ID,[['value',testAppHost]])
      else:
         testORBLocalhost_ID = AdminConfig.create('Property',testORB_ID,'[[name
com.ibm.CORBA.LocalHost] [value ' + testAppHost + ']]')
      showid(testORBLocalhost_ID)

   # All thread pools
   testPoolList = AdminConfig.list('ThreadPool',testServer_ID).splitlines()
   for testPool_ID in testPoolList:
      AdminConfig.modify(testPool_ID,[['inactivityTimeout',testInactivityTimeout]])
      showid(testPool_ID)

   # Testific thread pools
   for testPoolName,(testPoolMin,testPoolMax) in testPoolMap.items():
      testPool_ID = AdminConfig.getid('/Node:' + testNodeName + '/Server:' +
testServerName + '/ThreadPoolManager:/ThreadPool:' + testPoolName + '/')
      if testPool_ID != '':
         AdminConfig.modify(testPool_ID,[['minimumSize', testPoolMin], ['maximumSize',
testPoolMax]])
         showid(testPool_ID)

   # HTTP_2 inbound channel
   testHTTPInbound_ID = AdminConfig.getid('/Node:' + testNodeName + '/Server:' +
testServerName + '/TransportChannelService:/HTTPInboundChannel:HTTP_2/')
   AdminConfig.modify(testHTTPInbound_ID,'[[maximumPersistentRequests -1] [keepAlive
true] [readTimeout 6000] [writeTimeout 6000] [persistentTimeout 3000] [enableLogging
false]]')
   showid(testHTTPInbound_ID)

   # Transaction service
   testTransService_ID = AdminConfig.list('TransactionService',testServer_ID)
   AdminConfig.modify(testTransService_ID,'[[totalTranLifetimeTimeout 0]
[clientInactivityTimeout 0] [propogatedOrBMTTranLifetimeTimeout 0]]')
   showid(testTransService_ID)

   # JVM tuning
   testJVM_ID = AdminConfig.list('JavaVirtualMachine',testServer_ID)
   AdminConfig.modify(testJVM_ID,[['initialHeapSize', testJvmMin], ['maximumHeapSize',
testJvmMax], ['genericJvmArguments', testJvmArgs]])
   showid(testJVM_ID)
   if testMultiHome == 'false':
      testJVMMultiHome_ID = AdminConfig.getid('/Node:' + testNodeName + '/Server:' +
testServerName +
'/JavaProcessDef:/JavaVirtualMachine:/Property:com.ibm.websphere.network.useMultiHome/')
      if testJVMMultiHome_ID != '':
         AdminConfig.modify(testJVMMultiHome_ID,[['value',testAppHost]])
      else:
         testJVMMultiHome_ID = AdminConfig.create('Property',testJVM_ID,'[[name
com.ibm.websphere.network.useMultiHome] [value false]]')
      showid(testJVMMultiHome_ID)

   # Log tuning
   testLogList = AdminConfig.list('StreamRedirect',testServer_ID).splitlines()
   for testLog_ID in testLogList:
      AdminConfig.modify(testLog_ID,'[[rolloverType NONE] [maxNumberOfBackupFiles 1]]')
      showid(testLog_ID)

   # Trace service/trace log
   testTraceService_ID = AdminConfig.list('TraceService',testServer_ID)
   #AdminConfig.modify(testTraceService_ID,'[[enable false] [startupTraceTestification
"*=fatal"]]')
   #showid(testTraceService_ID)

   testTraceLog_ID = AdminConfig.showAttribute(testTraceService_ID,'traceLog')
   AdminConfig.modify(testTraceLog_ID,'[[rolloverSize 200] [maxNumberOfBackupFiles 20]]')
   showid(testTraceLog_ID)

   # PMI service
   testPMIService_ID = AdminConfig.list('PMIService',testServer_ID)
   AdminConfig.modify(testPMIService_ID,'[[enable false]]')
   showid(testPMIService_ID)

   # EJB cache
   testEJBCache_ID = AdminConfig.list('EJBCache',testServer_ID)
   AdminConfig.modify(testEJBCache_ID,'[[cacheSize 10000]]')

   # Setup install target
   if testTarget != '':
      testTarget += '+'
   testTarget += 'WebSphere:node=' + testNodeName + ',server=' + testServerName

# Install testj App
debug("\nInstall app to target: ",testTarget)
AdminApp.install(testAppEAR,'[-verbose -deployejb -deployejb.dbtype ' + testAppDB + ' -
deployws -usedefaultbindings -useMetaDataFromBinary yes -target ' + testTarget + ' ]')

import code; code.interact(local=locals())

#AdminApp.install('/opt/testj.ear','[-verbose -deployejb -deployejb.dbtype DB2UDB_V97 -
deployws -usedefaultbindings -useMetaDataFromBinary yes -target
WebSphere:node=Custom01Node,server=server1+WebSphere:node=Custom02Node,server=server2 ]')
```

ABOUT PRINCIPLED TECHNOLOGIES

Principled Technologies, Inc.
1007 Slater Road, Suite 300
Durham, NC, 27703
www.principledtechnologies.com

We provide industry-leading technology assessment and fact-based marketing services. We bring to every assignment extensive experience with and expertise in all aspects of technology testing and analysis, from researching new technologies, to developing new methodologies, to testing with existing and new tools.

When the assessment is complete, we know how to present the results to a broad range of target audiences. We provide our clients with the materials they need, from market-focused data to use in their own collateral to custom sales aids, such as test reports, performance assessments, and white papers. Every document reflects the results of our trusted independent analysis.

We provide customized services that focus on our clients' individual requirements. Whether the technology involves hardware, software, Web sites, or services, we offer the experience, expertise, and tools to help our clients assess how it will fare against its competition, its performance, its market readiness, and its quality and reliability.

Our founders, Mark L. Van Name and Bill Catchings, have worked together in technology assessment for over 20 years. As journalists, they published over a thousand articles on a wide array of technology subjects. They created and led the Ziff-Davis Benchmark Operation, which developed such industry-standard benchmarks as Ziff Davis Media's Winstone and WebBench. They founded and led eTesting Labs, and after the acquisition of that company by Lionbridge Technologies were the head and CTO of VeriTest.